The Dissertation Committee for Benjamin Shelton Kirk
certifies that this is the approved version of the following dissertation:

Adaptive Finite Element Simulation of Flow and Transport Applications on Parallel Computers

Committee:

Graham F. Carey, Supervisor

Clint N. Dawson

David S. Dolling

Robert J. MacKinnon

Hans M. Mark

Philip L. Varghese

Adaptive Finite Element Simulation of Flow and Transport Applications on Parallel Computers

by

Benjamin Shelton Kirk, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2007

For Lauren.

Acknowledgments

I would like to thank my advisor, Dr. Graham F. Carey, for his continued support and advice. I also wish to thank all of my colleagues in the CFDLab, most notably John Peterson, Dr. Bill Barth, and Dr. Michael Anderson, for many useful discussions at the white board. I am also grateful to the U.S. Department of Energy as this work was made possible in part by the Department of Energy Computational Science Graduate Fellowship. I also thank my current employer, the National Aeronautics and Space Administration, for providing me the freedom and flexibility to complete this work.

Mr. Joseph Coblish of the Arnold Engineering Development Center in White Oaks, Maryland provided much of the validation data used in the transient double-cone simulations presented in Chapter 5. Mr. Randolph Lillard of the NASA Lyndon B. Johnson Space Center aided this work by suggesting other high-quality hypersonic aerothermodynamic experimental data which were instrumental in validating the finite element formulation used in this work. Finally, my wife Shanee deserves thanks for her never-ending support in what surely seemed like a never-ending process.

Adaptive Finite Element Simulation of Flow and Transport Applications on Parallel Computers

Publication No. _____

Benjamin Shelton Kirk, Ph.D.
The University of Texas at Austin, 2007

Supervisor: Graham F. Carey

The subject of this work is the adaptive finite element simulation of problems arising in flow and transport applications on parallel computers. Of particular interest are new contributions to adaptive mesh refinement (AMR) in this parallel high-performance context, including novel work on data structures, treatment of constraints in a parallel setting, generality and extensibility via object-oriented programming, and the design/implementation of a flexible software framework. This technology and software capability then enables more robust, reliable treatment of multiscale–multiphysics problems and specific studies of fine scale interaction such as those in biological chemotaxis (Chapter 4) and high-speed shock physics for compressible flows (Chapter 5).

The work begins by presenting an overview of key concepts and data structures employed in AMR simulations. Of particular interest is how these concepts are applied in the physics-independent software framework which is developed here and is the basis for all the numerical simulations performed in this work. This open-source software framework has been adopted by a number of researchers in the U.S. and abroad for use in a wide range of applications.

The dynamic nature of adaptive simulations pose particular issues for efficient implementation on distributed-memory parallel architectures. Communication cost, computational load balance, and memory requirements must all be considered when developing

adaptive software for this class of machines. Specific extensions to the adaptive data structures to enable implementation on parallel computers is therefore considered in detail.

The `libMesh` framework for performing adaptive finite element simulations on parallel computers is developed to provide a concrete implementation of the above ideas. This physics-independent framework is applied to two distinct flow and transport applications classes in the subsequent application studies to illustrate the flexibility of the design and to demonstrate the capability for resolving complex multiscale processes efficiently and reliably.

The first application considered is the simulation of chemotactic biological systems such as colonies of *Escherichia coli*. This work appears to be the first application of AMR to chemotactic processes. These systems exhibit transient, highly localized features and are important in many biological processes, which make them ideal for simulation with adaptive techniques. A nonlinear reaction-diffusion model for such systems is described and a finite element formulation is developed. The solution methodology is described in detail. Several phenomenological studies are conducted to study chemotactic processes and resulting biological patterns which use the parallel adaptive refinement capability developed in this work.

The other application study is much more extensive and deals with fine scale interactions for important hypersonic flows arising in aerospace applications. These flows are characterized by highly nonlinear, convection-dominated flowfields with very localized features such as shock waves and boundary layers. These localized features are well-suited to simulation with adaptive techniques. A novel treatment of the inviscid flux terms arising in a streamline-upwind Petrov-Galerkin finite element formulation of the compressible Navier-Stokes equations is also presented and is found to be superior to the traditional approach. The parallel adaptive finite element formulation is then applied to several complex flow studies, culminating in fully three-dimensional viscous flows about complex geometries such as the Space Shuttle Orbiter. Physical phenomena such as viscous/inviscid interaction, shock wave/boundary layer interaction, shock/shock interaction, and unsteady

acoustic-driven flowfield response are considered in detail. A computational investigation of a 25°/55° double cone configuration details the complex multiscale flow features and investigates a potential source of experimentally-observed unsteady flowfield response.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Many physical processes of interest in engineering exhibit phenomena over a range of scales. Such multiscale behavior can result from complex interactions in reaction–diffusion systems, through the interplay of convection and diffusion in transport applications, or from other sources. The primary objective of this work is to investigate and advance adaptive finite element techniques and supporting software infrastructure for solving this class of problems, particularly on parallel computers, and to conduct basic science studies of complex fine-scale interaction using the simulation capability developed here.

1.1 Motivation

Multiscale physical processes are particularly challenging for efficient, reliable computational simulation using traditional mesh-based approaches. For such processes, the spatial resolution required for accurate simulation is intimately tied to the intricacies of the solution at hand, and hence a suitable mesh for an application is dependent upon the (generally unknown) details of the solution. Physical intuition has been the norm in practice for constructing a mesh a priori. This is the approach typically used to resolve boundary layer flows in steady external aerodynamic applications, for example. For the general case in which the location and structure of key features are not known until the approximate solution is obtained a reliable and near-optimal mesh for a given problem cannot be generated a priori. A user-in-the-loop approach to this problem leaves the analyst in the unenviable position of making decisions based on a mesh that may not provide adequate resolution and therefore yields erroneous results.

By contrast, adaptive techniques essentially obtain for the solution and the near-optimal mesh simultaneously. This approach removes inefficient manual mesh improvement/solution loop with an automated feedback control approach. In addition to providing an optimal mesh targeting the particular features of interest in a given problem (and thus minimizing computational resources required), adaptive techniques also provide a robust mechanism for efficient error control and faster, more stable solution algorithms.

It is now well accepted that adaptive techniques generally produce more robust and accurate simulation results. Still, while such techniques are often used in the research setting, their adoption for practical engineering computations is still not widespread. This is in no small part due to the difficulty posed by efficiently implementing these techniques, especially on modern high-performance parallel computing architectures.

Parallel computing platforms pose a particular challenge for efficient adaptive simulations and the necessary enabling software technology. The dynamic nature of adaptive schemes produce complications for domain decomposition strategies on parallel machines that have historically limited their application. A flexible, object-oriented software framework for adaptive finite element simulations on parallel machine was developed during the course of this work to address these difficulties.

1.2 Overview

Chapter 2 presents an overview of adaptive mesh refinement including the motivation and a historical perspective. This chapter introduces a number of concepts which will be recurring themes in subsequent application studies. Data structure requirements will be considered and a particular choice will be presented in detail. Finally, detailed aspects of AMR such as refinement strategies and error indicators will be discussed.

Chapter 3 discusses particular challenges which arise when adaptive methods are implemented on parallel computers. This chapter deals with some of the software engineering and design choices which must be made when implementing this class of methods.

These issues are addressed and the specific design and implementation employed in the `libMesh` finite element library is presented in detail. Of particular interest are new contributions to AMR in the context of parallel, high-performance computing, including novel work on data structures, treatment of constraints in a parallel setting, generality and extensibility via object-oriented programming, and the design/implementation of a flexible software framework. This framework is then applied in the specific study of fine scale interaction such as those in biological chemotaxis (Chapter 4) and high-speed shock physics for compressible flows (Chapter 5).

Chapter 4 considers chemotactic transport processes arising in biological systems such as *Escherichia coli* colonies to illustrate the flexibility of the framework for different physics classes and to explore adaptivity for chemotaxis for the first time. Such systems exhibit evolution on multiple time and spatial scales and require adaptive schemes for efficient simulation. A nonlinear reaction-diffusion model for this problem class is presented and a corresponding finite element formulation is developed.

In Chapter 5 the adaptive techniques are applied to a range of problems arising in compressible flows. For such flows the multiscale resolution capabilities of adaptive meshes are particularly well–suited to resolving localized features such as shock waves and boundary layers. In this chapter a finite element formulation for the compressible Navier–Stokes equations is presented and a number of application studies are performed.

Finally, Chapter 6 summarizes the work performed and discusses open issues which should be considered as future work.

1.3 Objectives

The primary goals of this work are to (1) develop and implement software algorithms and data structures for adaptive mesh refinement simulations, particularly in the context of high-performance parallel computers, and (2) to use these adaptive techniques to perform a range of application studies for problems arising in the disparate areas of

biological system modeling and in hypersonic aerothermodynamics.

1.3.1 A Software Framework for Parallel Adaptive Mesh Refinement Simulations

The current work addresses this aspect of adaptive simulation in detail and illustrates the viability of adaptive schemes on parallel platforms. During the course of this work the open-source finite element library `libMesh` was created and has served as a testbed for parallel adaptive finite element simulations across a wide range of physical disciplines. This software framework provides an extensible, object-oriented implementation the ideas explored in this work. All simulations presented in subsequent chapters were performed by the author using application codes built on top of this framework.

1.3.2 Application Studies

Chapters 4 and 5 apply the technology developed in the preceding chapters to two distinct physical applications. First, a nonlinear reaction-diffusion model which captures key biological processes involved in the pattern formation observed in chemotactic bacteria colonies is considered. Then laminar, supersonic/hypersonic calorically perfect gas flows in aerospace applications are simulated. Although these two problems come from distinctly different applications, there is similarity in that they both exhibit multiscale phenomena in the form of highly localized features, making them amenable to solution via adaptive techniques.

1.3.2.1 Biological Transport

Recently mathematical models have been developed which describe the local interactions which occur in bacterial populations to produce global patterns. These patterns often exhibit very localized, transient features which require extremely fine computational meshes to resolve. However, the transient nature of these features implies that static uniform high-resolution mesh simulations are inefficient as substantial computational

resources must then be devoted to portions of the domain which are devoid of features for the majority of the simulation. The case of complex patterns which are experimentally observed in *Escherichia coli* colonies are simulated in parallel using adaptive mesh refinement for the first time and detailed studies of localized chemotactic behavior are made.

1.3.2.2 Compressible Flows

The compressible Navier-Stokes equations for a laminar, calorically perfect gas are presented and discretized using a modified form of the streamline-upwind Petrov Galerkin finite element method. A novel treatment of the inviscid flux discretization is found to increase the stability of the method when using the popular conservation variables. This increased stability allows the method to be applied in hypersonic flows with strong shocks including shock/boundary layer and shock/shock interactions. These flows naturally exhibit multiscale behavior that can be captured efficiently on adapted meshes. The methodology is validated by comparison with experimental data including measured skin friction, pressure, and surface heat transfer for a number of configurations. Unsteady flows of an acoustic cavity and an axisymmetric double cone geometry are also considered. In the case of the double cone, numerical experiments are performed which support the conjecture that freestream noise is responsible for inducing wildly unsteady response observed in recent experiments conducted at low Reynolds numbers.

1.4 Contributions

1.4.1 Primary Contributions

The primary contributions of this work which will be described in the subsequent chapters are:

1. A new treatment of several algorithmic issues involved in performing adaptive flow and transport simulations on serial and parallel computers.

2. The development and implementation of a suite of flexible data structures for performing adaptive mesh refinement simulations on parallel computers.
3. A publicly available, physics-independent software framework for performing adaptive mesh refinement simulations [1].
4. A novel approach to computing algebraic constraints for adaptively refined meshes that exploits data locality and eliminates ad–hoc constraints on refinement levels admitted for neighboring elements [1].
5. The first known adaptive mesh refinement simulations of chemotactic reaction–diffusion biological systems and supporting detailed scientific simulation studies of evolving chemotactic patterns.
6. A modified inviscid flux discretization for high–speed compressible flows which enhances stability when employing the popular conservative variables.
7. Validation of a new software implementation for solving the compressible Navier-Stokes equations by comparing to high-quality experimental data.
8. An in-depth analysis and phenomenological studies of both steady and transient hypersonic laminar perfect gas flows focusing on:
   (a) Inviscid supersonic flow about a blunt body,
   (b) viscous/inviscid interactions,
   (c) acoustic cavity-induced oscillatory flow,
   (d) shock/shock interactions, and concluding with
   (e) three-dimensional hypersonic flows about reentry vehicles.
9. Development and testing of a new approach for implementing the no–penetration condition in the Euler equations in an implicit fashion. This approach is motivated by the weak formulation of the Euler equations and is trivial to implement on adaptively refined meshes.

1.4.2 Additional Contributions

The physics-independent adaptive capability developed in support of this work has been applied by the author to a number of other problem classes which will not be presented in detail in this work, including:

1. Density-driven porous media transport with applications in groundwater flows [2].
2. Rayleigh and Rayleigh-Bénard-Marangoni instabilities in incompressible flows [3].
3. The cascadic multigrid method applied to stationary incompressible flows [4].
4. Supersonic viscous flows through inlets in aerospace applications [5].

Chapter 2

An Adaptive Mesh Refinement Software Framework

Adaptive Mesh Refinement (AMR) allows for efficient numerical simulation because the spatial mesh resolution is optimally adapted in some sense for a given problem. This chapter will introduce key concepts and introduce data structures which are particularly well-suited for use in the AMR applications considered here. Chapter 3 will then address the implementation of adaptive methods on parallel computers. Finally, Chapters 4 and 5 will present application studies which apply the adaptive techniques discussed in this chapter.

The concepts discussed herein have been implemented concretely in the `libMesh` open-source software library. The library was initiated by the author to aid in (i) testing the ideas presented here and (ii) to enable the application of parallel AMR solution schemes to a wide range mathematical models for physical systems. The library itself is not tied to any particular application, and consequently is being adopted by a number of researchers in the U.S. and abroad for use in a wide range of applications.

2.1 Introduction

A primary goal of AMR is to enable more efficient and reliable numerical simulations. Some examples for problems of boundary layer type and nonlinear problems with singularities in [5] demonstrate the effectiveness of AMR in flow and transport simulations. In particular, AMR can significantly increase the range of problems that can be attempted given a limited set of resources on workstation-class (or larger) systems. However, it is also obvious that adaptive meshes imply a need for more general data structures. This, in turn,

leads to increased complexity, especially in parallel distributed AMR simulations.

In a typical adaptive refinement scheme, the solution on a given mesh is post-processed to obtain local error indicators that provide feedback for selective local mesh refinement [5, 6]. The basic approach for the simulation of partial differential equations in parallel with AMR applied to a steady state problem proceeds as follows:

1. First, an initial mesh is generated, the mesh is partitioned to subdomain meshes, and a solution is computed in parallel on this parent mesh (stabilization may be required for flow or transport in which convective effects are significant at this mesh scale). The details of the mesh partitioning scheme will be deferred until Chapter 3.

2. Next, a posteriori error indicators are computed from the approximate solution on the current mesh and a subset of cells are flagged by the error indicator.

3. These cells are subdivided and continuity constraints are enforced (in the subsequent solve step) at "hanging" nodes on element edges or faces shared with adjacent, unrefined elements.

4. If the load balance has not significantly changed as a result of the local refinement step, then repartitioning is not needed at this stage and the parallel solution over the existing partition continues with the new adapted mesh. Otherwise, repartitioning is carried out with due attention to the unbalanced tree data structure and the edge-neighbor information required on the partition interface as described below.

In solving evolution problems, the same general approach is applied within each time step. The solution and the adapted, partitioned mesh at the end of the previous time step become the starting solution and mesh for the next time step. Simulation proceeds as in the static AMR situation above except that now coarsening of previously refined cells also becomes more important due to the changing spatial behavior of the solution in time.

Note, in particular, that the steps outlined above are in general independent of *what* is actually being simulated. It is therefore possible to separate the software implementation which enables parallel simulations using adaptive mesh refinement from the physics-specific code required for a given application. This approach was taken as part of the present work by developing the `libMesh` library which in a sense amortizes the effort required to implement a parallel adaptive capability by allowing the supporting software infrastructure to be used for a wide range of problems. A similar approach has been applied most notably by researchers at Sandia National Laboratories in the `SIERRA` framework which is used for massively parallel simulations of multiphysics applications [7].

The remainder of this chapter discusses several key data structures which have been developed for use in the present work. An overview of object-oriented scientific computing software is first presented as this is the paradigm used to implement the relevant data structures. The discussion will focus on how these data structures can be used to perform adaptive mesh refinement simulations without regard to the particular computing paradigm used. The subsequent chapter will then consider specific issues which arise when this approach is applied on parallel, distributed memory computers.

2.2 Object-Oriented Scientific Computing

In recent years the performance of the C++ programming language has improved to the point that it provides a viable option for implementing high performance, object-oriented software. A complete discussion of object-oriented ideas and the features of a given programming language is outside the scope of this work. However, a cursory overview of object-oriented approaches (and their contrast to procedural approaches) is worthwhile as it provides the necessary background for the data structures described later in this chapter.

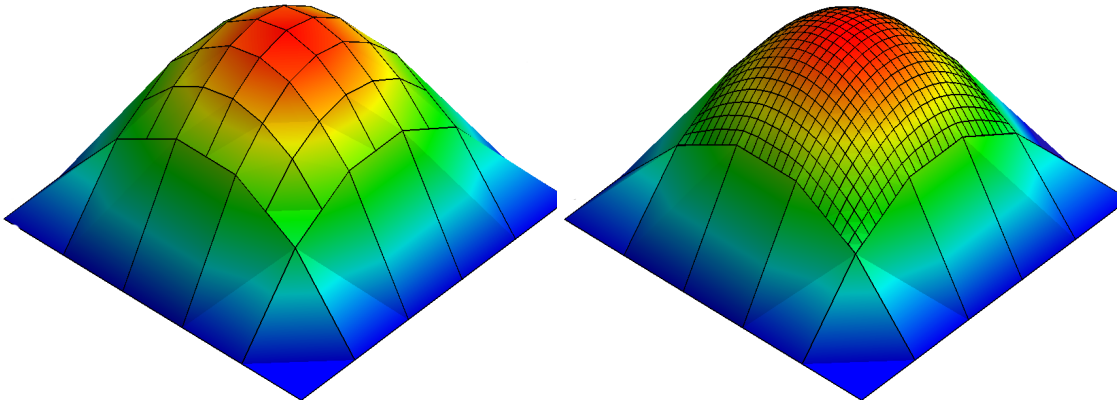
Two distinct paradigms for implementing software algorithms are procedure-oriented approaches and object-oriented approaches. The procedural approach has been the main-

stay of scientific computing for decades, dating back to the earliest versions of the `FORTRAN` programming language. In this approach basic computational kernels are constructed as a set of procedures which operate on some set of data. Implicit in this approach is an assumption as to the specific data types used to implement a given algorithm. One consequence of this is that the data storage and procedure implementation are intimately related. For example, suppose a standard array were used to store the individual elements of a vector. If for some reason it were decided that a linked-list would be a more efficient data structure, due to dynamic insertion and removal of elements for example, then substantial changes to all code which uses such a vector would be required.

By contrast, object-oriented approaches define specific *classes* which present characteristics and defined behaviors. An *object* is simply an instance of a given class. A significant benefit of objects is *encapsulation*, which is the ability to separate actual data from operations which are performed on the data. Returning to the vector example, in an object-oriented approach the specific data structure used to store numeric values could be completely encapsulated within an object, and code which uses such an object need not have any insight or access to this data structure. In this way encapsulation allows well-written objects to change algorithmic implementation or data storage techniques without affecting external code. For this and many other reasons, object-oriented programming is generally considered to create more maintainable and extensible software. Consequently, these approaches have become standard in many aspects of software engineering. The adoption of object-oriented techniques in the field of scientific computing has proceeded at a slow pace but is certainly gaining momentum.

2.3 Data Structures for Adaptive Mesh Refinement Simulations

Data structure selection is critically important for effective AMR implementation [1, 8]. In the results presented later, a tree structure is used. In this data structure each element can directly address both its "parent" and its "children." No assumptions are made here as to how many children an element may have. Additionally, each element can access its face



(a) one level of refinement mismatch. (b) three level of refinement mismatch.

Figure 2.1: Solution to a Poisson problem on meshes with one and three levels of mismatch at element interfaces.

neighbors. This data structure enables very flexible refinement strategies. For example, the familiar "level-one" restriction, in which adjacent elements are allowed to differ by only one level of refinement as shown in Figures 2.1(a) and 2.2(a), is not required by the present data structure. A depiction of a mesh which does not conform to the level-one rule is shown in Figure 2.1(b).

For continuous finite element approximation spaces, the finite element basis obviously must be continuous throughout the domain. Refinement interfaces like those shown in Figure 2.1 therefore require special treatment to ensure this continuity. The parent-child relationship depicted in Figure 2.2(b) is used in the present work to generate the required inter-element continuity constraints. This parent-child strategy is particularly useful on distributed memory machines since only local data is required. The general approach used in

(a) Level-one refinement with a hanging node

(b) Corresponding parent-child relationship

Figure 2.2: AMR data structures: neighbor and parent-child relationships.

this work is to constrain the child element to be continuous with a parent element. That is, on $\Omega_P \cap \Omega_C$:

$$\begin{aligned} u_C &= u_P \\ \sum \alpha_i \phi_i^C &= \sum \beta_i \phi_i^P \\ \boldsymbol{A}\boldsymbol{\alpha} &= \boldsymbol{B}\boldsymbol{\beta} \\ \boldsymbol{\alpha} &= \boldsymbol{C}\boldsymbol{\beta} \end{aligned}$$

Furthermore, since no assumptions are made about neighboring elements, this approach admits arbitrary refinement mismatch at element interfaces. This case of arbitrary element mismatch can be handled by introducing *recursive constraints*: If $\boldsymbol{u}_a = \boldsymbol{C}_b \boldsymbol{u}_b$ and $\boldsymbol{u}_b = \boldsymbol{C}_c \boldsymbol{u}_c$ then this implies $\boldsymbol{u}_a = \boldsymbol{C}_b \boldsymbol{C}_c \boldsymbol{u}_c$.

This approach has been used in the `libMesh` library to enable adaptive mesh refinement simulations using a wide variety of finite element types. Figure 2.3 shows the solution to the Poisson problem for an adapted hybrid mesh with mismatch in which the constraints were treated using this method. The cube base region is meshed with triangular prisms, and the pyramid region is composed of tetrahedra. The approach has recently been extended to the case of *hp*-adaptive finite element simulations in which the mesh is adapted

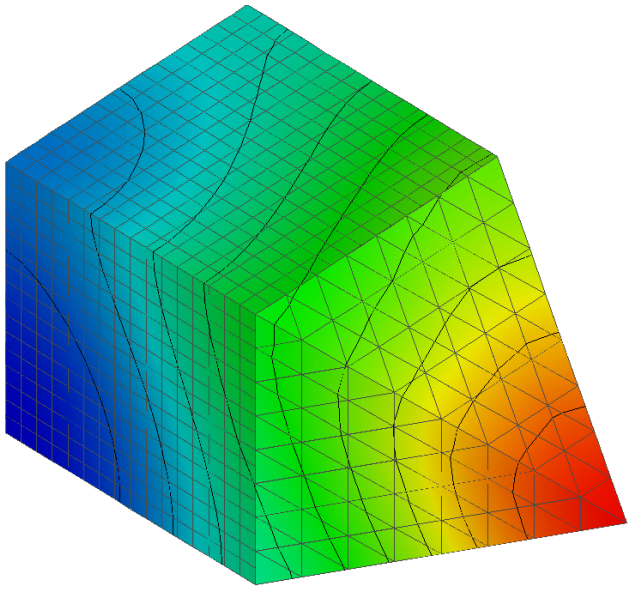


Figure 2.3: Adaptive mesh refinement on a three-dimensional hybrid grid.

spatially and the finite element approximation space is modified to provide enhanced convergence.

2.3.1 Mesh

The `Mesh` class is central to `libMesh` and was one of the first developed. It provides a discrete description of a d-dimensional object in D-dimensional space, where (d, D) are 1, 2, or 3. The class supports manifolds, so strictly speaking $d \leq D$. The discretization is composed of elements and nodes which are stored in the mesh, but the manner in which these data are stored is encapsulated by abstract classes which present implementation-independent interfaces to the user. This data encapsulation has allowed for re-factoring of the mesh class with minimal impact on the external application programming interface.

A base class/derived class structure is used to implement mesh input/output in various formats. Virtual base classes describe the interface for mesh input and output, and

derived classes provide the actual I/O functionality. The library currently supports reading and writing a number of unstructured mesh formats, including: the UCD format from AVS, the I-deas UNV Universal format, Exodus II from Sandia National Labs, GMSH, TetGen, Tecplot (ASCII and binary), and the GMV format from Los Alamos National Labs. The initial mesh is assumed to be conforming and provides the level-0 parent elements in the refinement hierarchy described in Section 2.3.4.3.

Custom iterator objects can be created to provide access to the elements and nodes contained in a mesh. The user can instantiate iterators to access all the elements in the mesh or some meaningful subset thereof. The latter approach is useful, for example, during parallel finite element matrix assembly on an adaptively refined mesh. In this case, the user obtains iterators which traverse the set of active elements (described in more detail in Section 2.3.4.3) which are owned by the local processor.

The mesh class is designed to be extensible. Encapsulating the stored elements and nodes by providing access only through custom iterators admits the possibility of providing different implementations for specific classes of meshes. The current `Mesh` implementation assumes a fully unstructured, hybrid element mesh. However, algorithmic and storage-based optimizations for Cartesian grids, block-structured grids, and grids with only a single type of element could be added without changing the current interface.

2.3.2 Degrees of Freedom

The first finite elements implemented in `libMesh` were the standard Lagrange elements with nodal value degrees of freedom. The library has since been extended to a wider variety of finite element types. Shape functions on more exotic finite elements can correspond to nodal Hessian components, mid-edge normal fluxes, or orthogonal hierarchic polynomials. With many shape functions there may be no single associated geometric point.

The `DofObject` class handles these different types of degrees of freedom generically. Examples of `DofObjects` are element interiors, faces, edges, and vertices. An element interior has associated degrees of freedom for those shape functions whose support is contained within the element. Face degrees of freedom correspond to shape functions contained within the two elements sharing a face, edge degrees of freedom correspond to shape functions for all elements sharing an edge, and vertex degrees of freedom correspond to shape functions supported on all elements sharing a single vertex.

2.3.3 Nodes

The `Node` class stores its (x, y, z) location in space, as well as additional state information including a unique `id` and its degree of freedom indices. The mesh data structure contains a complete list of all nodes. Nodes may be accessed directly by the user via iterators, or indirectly through elements which are connected to the nodes. Trivial operations which do not alter the mesh topology such as scaling, translating, or rotating a mesh are performed directly on the nodes.

During the refinement process new nodes may be added to the mesh. When two adjacent elements are refined, common nodes will exist on the inter-element interface. This situation must be properly resolved to achieve a valid discretization (i.e. with no duplicate nodes and associated "tears" in the mesh). A new node is created as a weighted combination of existing nodes. For each new node, a hash key is constructed based on these weights and global `ids` of its parent nodes. If this key already exists in the map of new node keys, the new node *may* be a duplicate and is therefore compared to any nodes with the duplicate key. (Note that if a "perfect hash" key were devised this comparison would be unnecessary.) Clearly, if the new node is a duplicate it is then rejected. This procedure relies on the near-uniqueness of the hash key and has been found to efficiently resolves nodal connectivity for refined elements.

Similarly, coarsening the mesh can create "orphan nodes," or nodes that are no longer connected to any elements. In an initial implementation these orphan nodes were

kept in place, so that they could be re-connected to future elements that may appear in the refinement process. In particular, in transient applications with some periodic behavior, elements will be created and destroyed repeatedly, and it was thought that leaving the nodes in place could speed up subsequent refinements. However, numerical experiments indicated that this approach did not provide an appreciable speedup, and the default behavior is now to remove such orphan nodes. After a refinement/coarsening step the library simply counts the number of elements connected to each node and removes those nodes with no element connections.

2.3.4 Elements

`libMesh` defines the abstract base class `Elem` which defines the interface for a geometric element. Concrete subclasses of `Elem`, such as `Quad4` and `Tet10`, are specialized via virtual function calls to return, for example, the correct number of nodes and sides when `n_nodes()` and `n_sides()` are called on an `Elem` pointer. The complete list of geometric element types provided in `libMesh` is shown in Figure 2.4. Note that an `Edge` *is* an `Elem` (in the polymorphic sense) in 1D, and similarly for `Face` in 2D and `Cell` in 3D. Implementations of all the standard geometric element types used in finite element analysis including quadrilaterals, triangles, hexahedra, tetrahedra, prisms, and pyramids, as well as a collection of infinite elements, are provided in `libMesh`.

2.3.4.1 Nodal Connectivity

Elements contain state information similar to nodes. Elements store a unique `id`, their processor `id`, and degree of freedom information. Additionally, the element connectivity is stored as pointers to nodes. This is a slight departure from the classic finite element data structure, in which the element connectivity is defined in terms of the nodal indices [9]. On 32-bit machines pointers and integers are both 4 bytes, so this choice does not impose additional storage. On 64-bit machines, however, pointers are 8 bytes, which essentially doubles the amount of memory required to store element connectivity.

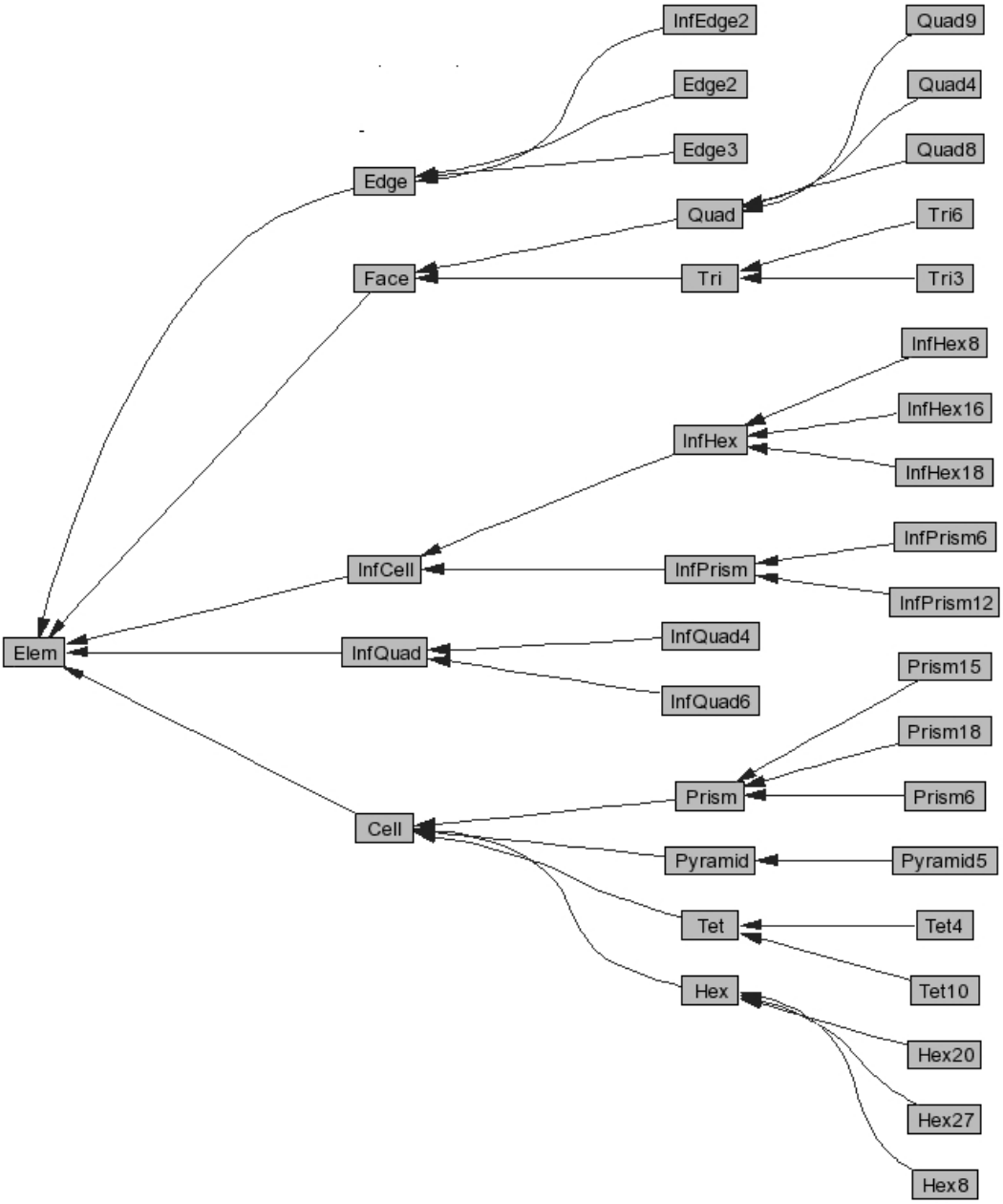


Figure 2.4: The `Elem` class hierarchy

This approach for storing the element connectivity was chosen so that elements could have increased functionality in the absence of a `Mesh` object. A traditional connectivity scheme would require the mesh to access the nodal locations of a given element. This is important, for example, when computing the map from a physical to reference element or determining if a point lies inside an element. By storing pointers to the nodes, the element can determine its geometric connectivity directly. This simplifies many functions in the code by requiring the user to pass only an element instead of both an element and the nodal locations. Additionally, this approach reduces the amount of indirect memory addressing required for an element to obtain nodal information as it may determine the physical location of its nodes with a single pointer dereference.

2.3.4.2 Face Neighbors

Elements also store pointers to their face neighbors. Two elements are said to be face neighbors if they share a "side," where a "side" is a node in 1D, an edge in 2D, and a face in 3D. If an element side is on the physical boundary of the domain there will be no neighbor. This is indicated in the code by a `NULL` pointer, so locating the elements coincident with the boundary is equivalent to finding all the elements with a `NULL` neighbor. This is useful, for example, when applying boundary conditions.

After reading a mesh from disk, or performing mesh refinement, it is necessary to construct the face neighbor information efficiently. The library handles this by looping over all the elements and then over the sides of the element. If a neighboring element has not been located already the side of the element is constructed and a hash key is computed based on the global indices of its nodes. A map is then queried to find any elements with sides matching this key, and they are checked for a possible match. The neighboring element will have the same side connectivity. When the matching neighbor is located it is removed from the map, therefore reducing the overall size of the map.

The loop through the N elements is $\mathcal{O}(N)$, while for a map of size M the lookup is $\mathcal{O}(\log M)$, so the resulting algorithm has $\mathcal{O}(N \log M)$ complexity. In the current imple-

mentation $M \leq N$, yielding a worst-case $\mathcal{O}(N \log N)$ algorithm. Alternate approaches are possible for which $M \ll N$ which could improve performance for very large meshes. For example, ordering the elements with a space-filling curve before performing the neighbor search will ensure adjacent elements are quickly located, reducing the maximum size of the map so that $M \ll N$.

Since constructing the side of an element is a common task, a special proxy class called `Side` has been developed for this purpose. This class essentially defines the side of an element as a new element living in a lower spatial dimension and provides the connectivity through a mapping from the original element. This approach allows the side of an element to be constructed rapidly, as the allocation and population of a new connectivity array is not required.

2.3.4.3 Element Refinement Hierarchy

Elements are refined upon user request via the "natural refinement" scheme. In this approach d-dimensional elements are generally refined into 2^d subelements of the same type. (Pyramid refinement is an exception to this rule: refining a pyramid results in a collection of pyramids and tetrahedral elements.) Hanging nodes are allowed at element interfaces and hanging degrees of freedom are constrained algebraically using the technique discussed previously. This approach was chosen because it is applicable for general hybrid meshes with arbitrary types of elements, and in general results in refined elements of the same type. This latter point ensures that refining an all-quad mesh in 2D produces an all-quad mesh, for example. Additionally, this approach results in refined elements which are geometrically similar to the parent, thereby preserving the initial element quality.

This refinement approach naturally yields a tree-like data structure. Figure 2.5 shows the quad tree-data structure which results from refining a single quadrilateral element. Each element has a pointer to its "parent," and an array of pointers to its "children." The initial, level-0 elements are unique in that they have no parent. This is indicated in the code with a `NULL` parent pointer. Similarly, the active elements which are used in finite el-

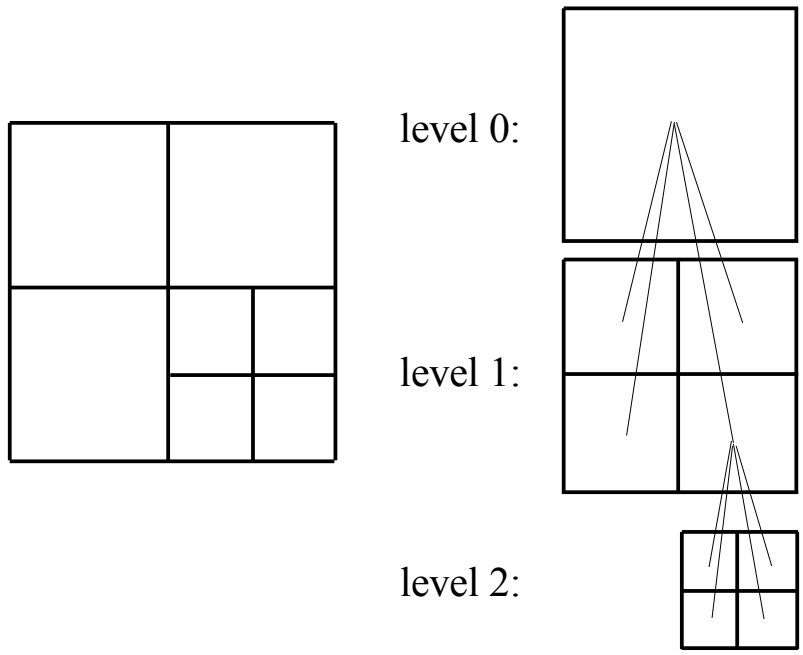


Figure 2.5: Element refinement hierarchy for a 2D quadrilateral mesh.

ement computations have no children, hence they have a `NULL` array of children. The level of a given element may be determined recursively from its parent. The user is allowed to access any subset of the elements via iterators as discussed previously. The active "leaf" elements are commonly used in matrix assembly, but intermediate levels could also be used in a multigrid cycle, for example.

The element hierarchy is additionally used to locate hanging nodes in the mesh which must be constrained. As mentioned previously, elements store pointers to neighboring elements which share sides. These neighboring elements are necessarily at the same level of refinement. If an active element neighbors a refined element (also referred to as an "inactive" element) then any degrees of freedom located on the common side must be constrained.

The refinement hierarchy also naturally supports element coarsening. In the case that all of the children of an element are flagged for coarsening, the parent element simply

deletes its children, `NULL`s its children array, and becomes active again. (Note that the neighbor connectivity of the parent is unchanged in this process and therefore does not need to be regenerated.) In Figure 2.5, this would correspond to all the level-2 elements being deleted. The resulting mesh would contain just the active level-1 elements and their parent. A consequence of this approach to element coarsening is that the mesh cannot be coarsened below the initial, level-0 mesh. In many cases it is desirable to use the coarsest level-0 mesh possible and allow the refinement process to add elements only where they are needed.

Finally, the refinement tree can be exploited when enumerating boundary conditions. A data structure is provided which maps an element and its relevant side(s) to a boundary condition enumeration. (Note that, in keeping with the physics-independent nature of the library, imposing the boundary conditions is left to the user.) By nesting the children such that boundary "sides" are coincident with the same sides as the parent, this data structure can be static and reused regardless of mesh level. This is implemented by recursively returning the parent's boundary condition information for the relevant side until the top-level parent is located and the map is ultimately queried.

Remark: The element concept discussed here corresponds to the geometric entities use in a discretization. This is a distinct notion within the library from the finite element approximation space used in the numerical simulation. This distinction allows for a convenient separation of the geometric modeling function of *elements* and the approximation properties of associated *finite elements*. For example, a two-dimensional simulation using piecewise linear Lagrange basis functions may use triangular and quadrilateral elements, but the underlying finite element type is specified by the use of a first-order Lagrange basis.

2.3.5 Systems

The abstract `System` class in `libMesh` corresponds to a set of one or more partial differential equations which are to be solved on a given mesh. `libMesh` provides several concrete system implementations including explicit, implicit, steady, transient, linear, and

nonlinear systems. A system stores the solution values for the degrees of freedom in a simulation, which may be either real or complex valued. Additionally, a system may contain additional information (such as a sparse matrix) required for a particular solution strategy. In the current implementation a system is uniquely tied to a given mesh, so a simulation that uses multiple meshes must also solve multiple systems.

The `System` class provides a generic, customizable interface which allows the user to specify the physics-dependent parts of an application. For example, in the case of an implicit system, users can provide a function for matrix assembly or can derive their own class and overload the matrix assembly operator. Similarly, for transient systems, the user may either provide an initialization function or overload the initialization operator provided in the library.

Multiple systems may be tied to a given mesh to allow for loose coupling of different physics. This feature has been applied to iteratively decouple the incompressible fluid flow and heat transfer equations. In this example two implicit systems are solved in an iterative fashion. Similarly, incompressible flows using pressure projection operator splitting techniques have been solved using a combination of loosely coupled explicit and implicit systems.

The library makes extensive use of C++ templates to allow complex systems to be constructed from simpler subsystems. For example, transient nonlinear systems are supported by combining a transient outer loop with a nonlinear inner loop. Templates are useful in this setting because they allow simple components to be combined into a complex algorithm. This leads to code reuse and minimizes debugging efforts.

2.4 Finite Element Type Independence in Adaptivity

A primary goal of `libMesh` is extensibility: it should be easy for experienced users to add new finite element types to the system with minimal effort. To make this possible, `libMesh` includes implementations for hanging node constraints, solution restrictions to

coarsened meshes, and solution projections to refined meshes which are independent of the underlying finite element space. When adding a new finite element to the library, developers can first use these default implementations, only replacing them with optimized element-specific implementations if necessary for efficiency.

When using the hierarchical mesh refinement capabilities provided by `libMesh`, the resulting meshes are non-conforming, with "hanging nodes" on sides where coarse elements and more refined elements meet. The approach described previously for generating these constraints is generally applicable to arbitrary element types.

A subset of elements is selected for refinement based on some specified criteria. A new set of elements is constructed from these "parent" elements through a linear map. That is, for a given parent, its children are constructed dynamically through an "embedding matrix" which provides the required map. The inverse process, in which elements are selected for coarsening, will remove all the children of a given element and thus re-activate the parent. Both the coarsening and refinement processes alter the approximation space associated with a given finite element discretization.

Mesh coarsening will require the restriction of solution data onto a coarse parent element based on the approximate solution on its refined children. Similarly, mesh refinement will require the projection of solution data onto refined child elements from their original, coarse parent. The restriction and projection operator should be accurate, computationally efficient, uniquely defined, parallelizable, and independent of finite element type. Hilbert space projection operators are used to maintain the required element-type independence. Using an element-wise L_2 or H^1 projection is efficient, runs in parallel without interprocessor communication, and gives an exact solution in the case of refinement using nested finite element spaces. For coarsening, however, an element-wise H^s projection would not be uniquely defined, as the projections from neighboring cells could produce different function values along their shared side. A more complicated but similarly efficient algorithm restores uniqueness by acting on these shared degrees of freedom first, as follows:

We start by interpolating degrees of freedom on coarse element vertices. Holding these vertex values fixed, we do projections along each coarse element edge. Because these projections involve only data from the original refined elements on that edge and not data from element interiors, they are uniquely defined. In 3D, element faces are then projected while holding vertex and edge data fixed. Finally, element interior degrees of freedom are projected while holding element boundary data fixed. Although this series of projections is more complicated than a single per-element projection, the number of degrees of freedom to be solved for at each stage is much smaller (due to the dimensionally hierarchical nature of the approach), so the dense local matrix inversions required are faster.

2.5 Error Indicators and Refinement Criteria

The error indicators that guide AMR schemes are often based on element residuals, recovery techniques, local boundary value solves, and related approaches supported by an a posteriori error analysis to provide the underlying mathematical framework. Because of the nature of the operators and the importance of shock physics in later applications an inexpensive gradient or flux-jump error indicator η_K^{FLUX} of the form [10]

$$\eta_K^{\text{FLUX}} := \left(\frac{h_K}{24} \oint_{\partial K} |R_K|^2 ds \right)^{1/2} \tag{2.1}$$

is used, where R_K is the local residual defined by

$$R_K := \begin{cases} 0, & s \in \partial K \cap \Gamma_D \\ g_N - \nabla u_h \cdot n_K, & s \in \partial K \cap \Gamma_N \\ \frac{1}{2}(\nabla u_h|_L - \nabla u_h|_K) \cdot n_K, & s \in \partial K \cap \partial L \neq \emptyset \end{cases} \tag{2.2}$$

and where g_N is given Neumann boundary data, n_K is the outward unit normal for cell K of representative size h_K, and cell L shares an edge in two dimensions (or a face in three dimensions) with cell K in the finite element mesh.

Once the error for each element in the mesh is computed some selection criteria must be applied to determine the fraction of elements to be coarsened or refined. Several selection schemes are available in the `libMesh` library, including:

1. Selecting elements for refinement which exceed some specified fraction of the maximum error in the domain. Similarly, elements below some other (smaller) fraction of the maximum error are considered for coarsening.
2. Select some fixed fraction of elements for coarsening and refinement. This approach is attractive because the number of elements added in an individual refinement step may be bounded.
3. Select elements for coarsening and refinement by considering the mean and standard deviation of the error distribution.

It is this final strategy that is used in this work. Due to the highly localized features in the flow and transport problems the error itself may not form a normal distribution, but rather is assumed to be log-normal. This approach is motivated by the work of Aftosmis and Berger for the case of the three-dimensional, steady Euler equations in gas dynamics [11]. The mean (m) and standard deviation σ of the resulting distribution are computed. User-specified refinement and coarsening fractions, f_r and f_c, are used to select elements for adaptation. That is, any element whose log-error exceeds $(m + f_r\sigma)$ will be refined (up to a user-specified maximum level). Similarly, elements whose log-error is less than $(m - f_c\sigma)$ will be considered for coarsening. As mentioned previously, elements may be coarsened to the level of the initial mesh but no further. Thus it is often beneficial to use very coarse background meshes when possible to allow the coarsest possible discretization in benign regions of the domain.

Summary

An overview of adaptive mesh refinement algorithms has been presented and a suite of data structures which have been implemented in the `libMesh` library have been examined. Up to this point the focus has been on general issues associated with adaptive schemes and has not considered the underlying computer architecture on which the methodology is ultimately implemented.

In the following chapter, specific issues which arise when implementing adaptive schemes on distributed-memory parallel architectures are discussed. Extensions and data dependencies for the data structures presented in this chapter are addressed. The current level parallelization in the `libMesh` library will be discussed and extensions for fully parallelizing the library will be examined.

In subsequent chapters, the resulting software technology is applied to two distinct physical problems. The first application class considered is that of chemotactic bacteria colonies. Compressible, high-speed viscous and inviscid flows of interest in aerospace applications are then investigated. In both cases, the mathematical models employed, discretization approach, and solution scheme employed will be analyzed in detail. A range of applications is then considered within each problem class to assess the suitability of the adaptive simulation approach.

Chapter 3

Parallel Issues and Data Structures

Adaptive simulation techniques necessarily produce dynamic simulations which have historically been difficult to implement effectively on parallel computers. This chapter continues from the AMR software framework considerations of Chapter 2 to address particular issues which arise on parallel, high-performance computers. We begin with a brief overview of parallel architectures and a discussion of their importance. Next the particular challenges for AMR schemes in this setting are described, along with a discussion of how particular data structures may be used to help ease the implementation of adaptive methods on parallel architectures.

3.1 Parallel Computing

3.1.1 Background

Parallel computing has become the *de facto* method used to achieve high-performance, large-scale simulation. These parallel machines have largely replaced specialized vector machines for scientific computing, due in part to their price per-performance benefit.

In the 1970's and '80's scientific computing represented a significant portion of the computer market. Specialty machines which used vector processing units, produced by companies such as IBM and Cray, were common. From the mid-1990's to present day, however, the landscape has changed remarkably. Scientific computing now represents only a small fraction of the computer market. Rather than driving technology innovations, the scientific computing market now finds itself adapting hardware developed largely for other fields.

Parallel computing is one way in which scientific computing has dealt with this changing landscape. In general, the distinguishing feature of a parallel architecture is that each processor is connected to its own memory resources which cannot be seen by other processors. This *distributed memory* paradigm introduces particular complications in the development of parallel algorithms, as some form of message passing must be used to enable communication between collaborating processors. This is in contrast to *shared memory* machines, in which each processor has access to a large, global pool of memory. The two basic architectures are represented schematically in Figure 3.1. Parallelizing software for shared memory architectures is simpler than in the distributed case because data can be immediately accessed by all processors.

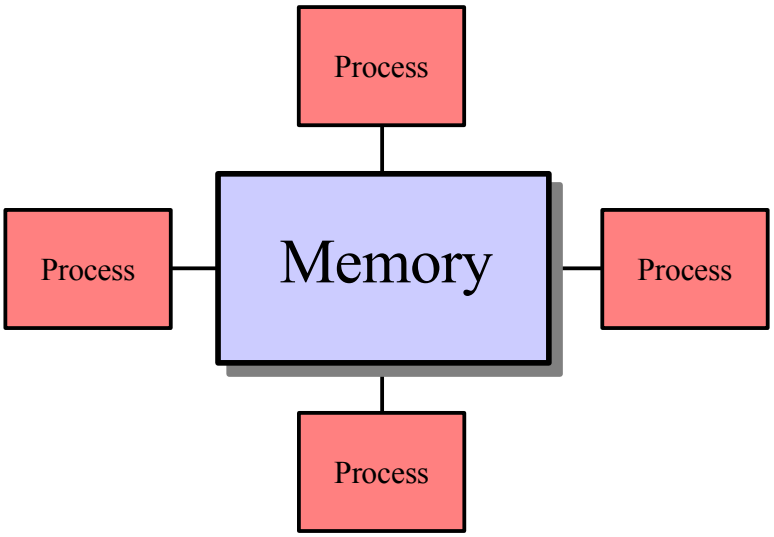
The parallel architecture paradigm provides one strong benefit: scalability. The vast majority of the world's fastest computers as of the time of this writing are massively parallel machines comprised of $\mathcal{O}(10,000)$ commodity server-class machines.[1]

3.1.2 Implications for Mesh-based Simulations

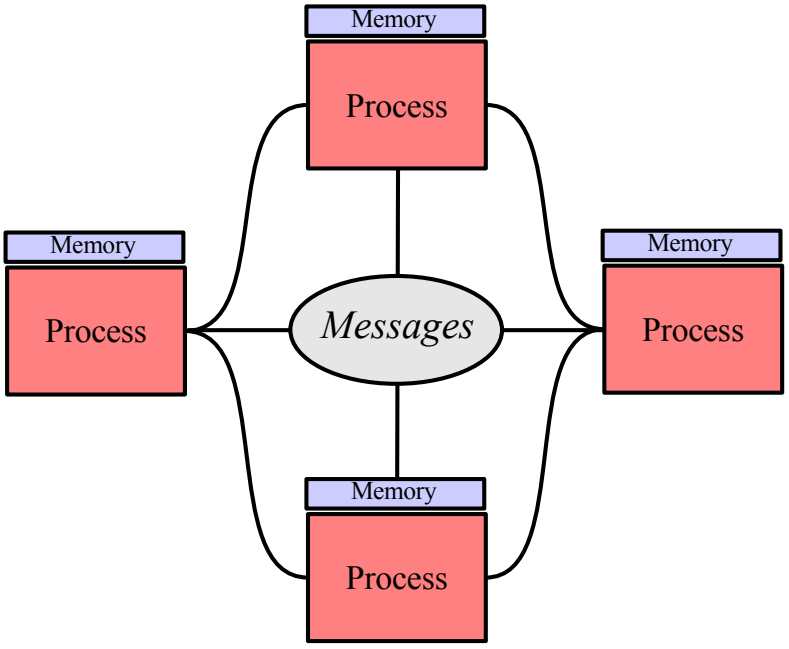
Mesh-based simulation techniques approximate the solution to a set of governing equations by solving a discrete representation of the problem. Discretization is performed with a mesh that describes the physical domain of interest. For parallelization, a computational mesh can be decomposed into a number of subdomains which are each assigned to one of the processors used in the simulation. This *domain decomposition* approach is common and will be the focus of Section 3.2.

Of course, some form of communication is required between processor subdomains in this approach. The amount of data required to be communicated between processors to enable this approach is highly dependent on the partitioner used to create the domain decomposition and on the choice of data structures used in the software implementation, as this drives which data are required from adjacent processors. Some key aspects of the data

[1] `http://www.top500.org`, January 2007.



(a) shared memory



(b) distributed memory

Figure 3.1: Shared and distributed-memory parallel architectures.

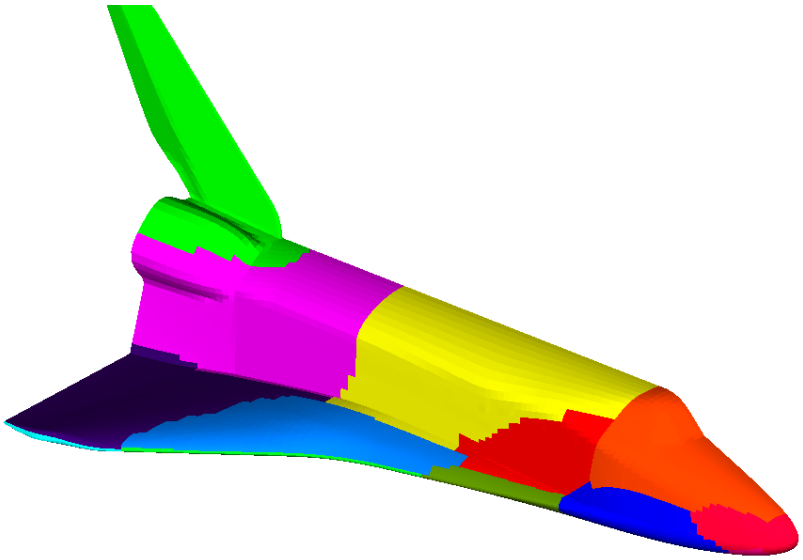


Figure 3.2: Element-based domain decomposition of a surface mesh into 16 subdomains

structures used in this work will be discussed in Section 3.3 with an emphasis on how they influence the parallel implementation.

In general, implicit techniques require that a linear system be constructed and solved as part of the solution algorithm. In parallel the construction of this system is largely unchanged from the serial case. The solution of the system, however, must account for the distributed nature of the machine. Specifics regarding parallelization within finite element simulations is the focus of Section 3.4.

3.2 Domain Decomposition

A non-overlapping domain decomposition approach is used in `libMesh` to achieve data distribution on parallel computers as shown in Figure 3.2 [6]. The discrete domain Ω_h is partitioned into a collection of subdomains $\{\Omega_h^p\}$ such that $\bigcup \Omega_h^p = \Omega_h$ and $\bigcap \Omega_h^p = \emptyset$. Each subdomain Ω_h^p is assigned to an individual processor. Two primary metrics in judging the quality of a partitioning scheme's computation and communication load balance are (1)

the subdomain mesh size and (2) the number of "edge cuts" in the resulting partition. For a mesh composed of a single type of element, each subdomain should contain essentially an equal number of elements so that the computational work is load balanced across all available processors. The edge cut metric, on the other hand, is related to the interprocessor communication required by the parallel solver. For an overview of several domain decomposition strategies that are available see [12, 13].

In problems with high-resolution static meshes, the partitioning is only performed once. In such cases, a high-quality partition which simultaneously minimizes both the subdomain's size and edge-cut metrics may be desirable, even though it may be relatively expensive. For adaptive refinement and coarsening applications where the steady-state solution is of interest, it is frequently the case that one begins with a coarse mesh at the root level and selectively refines towards a near-optimal mesh with little coarsening. It is obvious that an initially balanced partition may become unbalanced very rapidly, leading to parallel computational inefficiencies. Consequently, the mesh typically requires frequent repartitioning during the AMR process. In this case a less expensive partitioning scheme may be desirable as it will be used repeatedly. The development of optimal schemes for repartitioning that can take advantage of a prior partition in a parallel AMR setting is still an open research issue [12].

In `libMesh` we partition by default using the recursive scheme provided by METIS for coarse parallel granularity when the number of selected partitions $n_p \leq 8$, and with the k-way scheme otherwise [14]. A space filling curve partitioning algorithm is also available, as is an interface to ParMETIS. The frequency of repartitioning needed will, in general, depend on the evolving imbalance. Currently, repartitioning is done every time the mesh changes (i.e. every time refinement or coarsening occurs). Profiling suggests that this technique is not overly inefficient for typical applications, but it could be very slow for large-scale problems, and clearly it is unnecessary if the refinement scheme selects only a small number of elements to be refined and coarsened.

Another issue that must be considered is the subset of the AMR tree on which the partitioning algorithm acts. Typically, the partitioning algorithm is applied to all the active elements (i.e., the leaves of the AMR tree) so that the resulting computational load is well balanced. However, this approach involves calling the partitioning algorithm on a large subset of the AMR tree, while it may be sufficient to partition based on a coarser level and simply assign all the children of these coarse level elements to the same processor. Due to the parallel implementation of the `Mesh` discussed in Section 3.4, we do not (yet) consider the possibility that accessing an ancestor element would require off-processor communication. In such a scenario, one would need to ensure that repeated refinement and coarsening of the same region of the mesh did not lead to excessive communication overhead, perhaps by ensuring that a local synchronized copy of an element's parent is always available.

3.3 Data Structures – Parallel Aspects

This section describes several of the key `libMesh` data structures which are important for parallel implementation. The discussion focuses on basic functionality, possible extensions, and the reasoning behind certain design decisions in the context of parallel computing. Algorithms that are central to the library's functionality are also described.

3.3.1 Element Hierarchy

The refinement hierarchy and recursive constraint application procedure described in Section 2.3 was chosen because it exploits data locality and minimizes the amount of inter-processor communication required to generate element constraints.

In general, the refined mesh will no longer be well-balanced across all processors, and dynamic repartitioning (and possibly data redistribution) is required. The frequency of the dynamic repartitioning process will influence the parallel performance of the application. The optimal repartitioning frequency will depend on the relative compu-

tation/communication cost of the parallel system and the solution algorithm employed. To minimize the computational overhead incurred it is desirable that the entire refinement tree for a given root element reside in local memory.

3.3.2 Degrees of Freedom

The domain decomposition approach described earlier assigns disjoint groups of elements to individual processors. This allows the element-based degrees of freedom to be assigned uniquely to the processor which owns the element, but requires some shared distribution of vertex, edge, and face degrees of freedom which reside on subdomain boundaries. Figure 3.3 illustrates the approach which is used in the library.

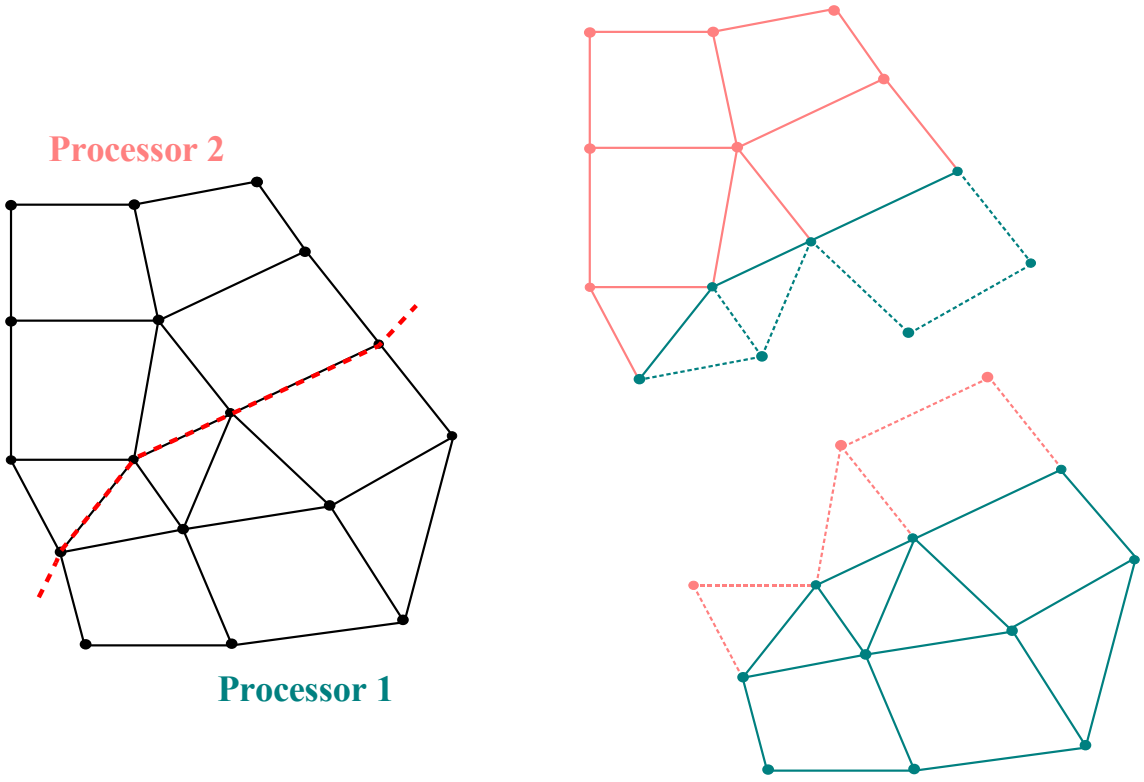


Figure 3.3: Element partitioning & degree of freedom distribution

In this approach, any degrees of freedom on the border between subdomains are owned by the processor of lowest global rank. This is evident from the figure, where the nodes on the shared interface have been assigned to processor 1.

This approach for assigning degrees of freedom to processors also fits well with the sparse matrix partitioning scheme employed in the underlying linear solver, in which complete rows of the sparse matrix are assigned to individual processors [15]. This is the natural matrix decomposition that results from the degree of freedom distribution depicted in Figure 3.3.

3.4 Parallelization in Finite Element Simulations

Parallelism in `libMesh` is present at the matrix assembly and linear algebra levels. On distributed memory machines, such as PC clusters, a complete copy of the mesh is maintained independently on each processor. This design decision currently limits practical 3D applications to the order of 128 processors because of the storage overhead associated with storing the global mesh. Nevertheless, a remarkable number of 3D applications have been successfully solved using this implementation. Fully parallel mesh data structures must contend with load balancing issues, however these issues are mitigated by keeping a copy of the mesh on each processor. The recent proliferation of hybrid distributed/shared memory architectures, such as PC clusters with multi-core CPUs, suggests that corresponding parallel codes should include combined message passing and multithreading models. While a complete parallelization of the global mesh is not attempted in the present work, Section 3.5 will outline the approach which is intended to be pursued as future work.

One major goal of the library is to shield end-users from the complexity of parallel programming, allowing them instead to focus on the physics they are modeling. The vision is that users develop and debug applications on serial machines and then move seamlessly to parallel architectures for large-scale simulations. To achieve this goal the library hides

parallel communication from the user, so basic message passing calls, for example, are not required at the user-level in most applications.

A case in point is the simple act of reading a mesh from disk. The user simply instantiates a `Mesh` object and calls its `read()` member function. This is a trivial operation from the user's point of view, consisting of only two lines of code. These two lines of code are then executed on every processor in a parallel simulation, causing a chain of events in which processor 1 actually reads the mesh from disk and broadcast the data to the remaining processors. This level of abstraction is common in many numerical libraries (e.g. PETSc) and provides encapsulation of the library implementation so as not to affect user code.

3.4.1 Data Dependencies

The parallel degree of freedom distribution discussed in Section 3.3.2 allows for shared degrees of freedom on processor boundaries. This allows local elements to both depend on and contribute to remote degrees of freedom. Hence, we will require some synchronization process to obtain remote data.

For a classic finite element discretization, computations on a given element are dependent solely on the element's own degrees of freedom. Synchronizing only the shared degrees of freedom is sufficient in this case. However, certain error indicators and discontinuous Galerkin schemes compute the interface flux jump between adjacent elements, which depends on all the degrees of freedom in a neighboring element. For this reason `libMesh` synchronizes not only shared degrees of freedom but all the degrees of freedom corresponding to the face neighbors of the local elements. This corresponds to all the degrees of freedom for the "ghost" elements depicted in Figure 3.3.

Synchronization is performed in the library after the completion of a solve step. For example, the completion of a linear solve will result in updated degrees of freedom on each processor, and a communication step is required so that updated values for remote degrees

of freedom are obtained on each processor. The library performs this step at the end of each solve without user intervention.

3.4.2 Matrix Assembly

The domain decomposition approach used in the library naturally lends itself to parallel matrix assembly. The matrix assembly code provided by the user simply operates on the active elements local to each processor. The standard approach of assembling element matrices into the global matrix for an implicit solution strategy is used. In this approach the data needed to assemble the local element matrices is collected before the assembly procedure, and the actual matrix assembly can be performed in parallel.

The degree of freedom distribution used in the library permits local element matrices to contribute to remote degrees of freedom for elements which intersect inter-processor boundaries. Hence, communication may be required in forming the global matrix. In PETSc (the underlying parallel linear solver used in this work), sparse matrix objects accumulate entries that must be communicated during the matrix assembly phase and cache them, which prevents costly inter-processor communication for each element in the assembly loop. After each element matrix is inserted on a given processor, communication is required to correctly sum the entries for these shared degrees of freedom. The matrix assembly phase can be summarized by the following steps:

1. Synchronize data with remote processors. This is required so that any remote data needed in the element matrix assembly (required in nonlinear applications, for example) is available on the local processor.
2. Perform a loop over the active elements on the local processor. Compute the element matrix and distribute it into the global matrix.
3. Communicate local element contributions to degrees of freedom owned by remote processors.

The first and third steps are performed automatically by the library, while the second step requires user-supplied code for forming the element matrices, or for residual evaluation in the case of Jacobian-free Newton-Krylov methods.

3.5 Mesh Parallelization Strategy

As mentioned previously, the current software implementation stores a copy of the global mesh on each processor involved in the simulation. Clearly, this approach is not scalable to parallel architectures with many hundreds to thousands of processors. At the time of this writing, the largest known simulation performed with the `libMesh` software used 400 processors. The goal of this section is to outline the approach that is envisioned for parallelizing the unstructured mesh data structures which are employed in the current implementation. This strategy has evolved from the inception of the library and, indeed, many of the current data structures were designed to extend naturally to the parallel implementation.

3.5.1 Static Mesh Parallelization

For the case of an unstructured mesh which is decomposed into a number of subdomains for each processor in a given simulation, parallelization may be accomplished by simply storing a subset of the global mesh on each processor. Clearly this subset will include the subdomain assigned to the processor but could also include other regions of the global mesh as required to satisfy data dependencies or facilitate other functionality in the library.

Returning to Figure 3.3 we see graphically the minimum amount of data which must be stored on each processor in the current scheme. This includes all of the elements assigned to the particular processor as well as any of their face neighbors which may be assigned to other processors. These "ghost" elements are necessary for certain functions such as locating refinement interfaces where degrees of freedom need to be constrained. It

should be noted that these elements are not used in any computations, they exist simply to complete the chosen data structures.

3.5.2 Degree of Freedom Indexing

When the global mesh is stored on each processor the task of indexing the global degrees of freedom is trivial. This process is outlined in Algorithm 3.1. All indices are

Algorithm 3.1 Indexing degrees of freedom for the case when the global mesh is stored on each processor.

```
1:  for p = 1 to N_processors do
2:    for e = 1 to N_local elements do
3:      for n = 1 to N_element nodes do
4:        if nodal degrees of freedom are not indexed
5:          index degrees of freedom
6:        endif
7:      end
8:      index element degrees of freedom
9:    end
10: end
```

initialized to some invalid number (to indicate that a degree of freedom index has yet to be assigned) and then each subdomain in the mesh is traversed element-by-element. Each element loops over its nodal degrees of freedom and assigns a global index to those degrees of freedom which have not already been visited. The same procedure is then repeated for the element-based degrees of freedom.

In parallel, however, this approach is problematic at inter-processor boundaries. For example, processor 1 can number its degrees of freedom in the usual way, but this poses a problem for processor 2. There is an inherit serialization step in that the global degree of freedom indices for a given subdomain depend on all the subdomains of lower index.

A parallel algorithm can be constructed from 3.1 by introducing intermediate local degree of freedom indices and extending the concept of element ownership to nodes. Just

as elements are assigned to a given processor, nodes may be as well. Of course, any node will need to be duplicated on processors which contain an element which is connected to it. By convention, a node is said to be owned by the processor of minimum rank p which has an element connected to that node. This is convention is shown schematically in Figure 3.4 for the case of three processors in two dimensions.

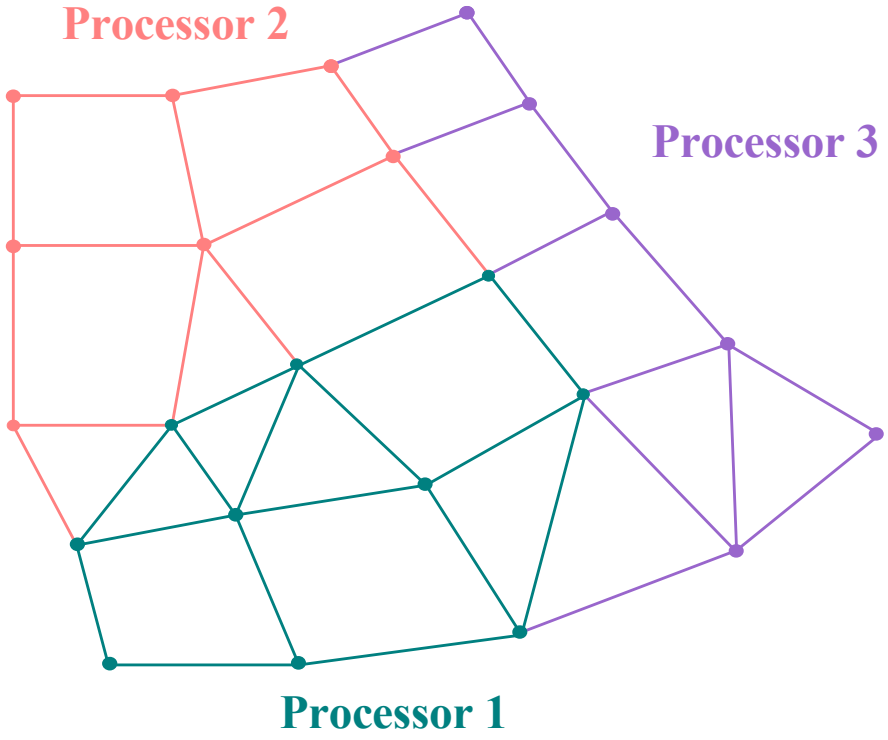


Figure 3.4: Element and node ownership for an unstructured mesh partitioned into three subdomains.

Parallel degree of freedom indexing can then be performed using the approach outlined in Algorithm 3.2. In this algorithm local degree of freedom indices are created for each processor in the range $[0, N^p_{dofs})$. All processes then broadcast the number of local degrees of freedom. The global degree of freedom offset for each processor is then computed, and the global degree of freedom indices are given as $[N^p_{off}, N^p_{dofs} + N^p_{off})$.

Algorithm 3.2 Indexing degrees of freedom for the case when the mesh is parallelized across all processors.

1: **for** $e = 1$ to $N_{\text{local elements}}$ **do**
2: **for** $n = 1$ to $N_{\text{element nodes}}$ **do**
3: **if** node is owned by p and degrees of freedom are not indexed
4: index degrees of freedom
5: **endif**
6: **end**
7: index element degrees of freedom
8: **end**
9: gather the number of degrees of freedom for all processors $\{N^p_{dofs}\}$
10: compute degree of freedom offset for processor p as

$$N^p_{off} = \sum_{i=1}^{p-1} N^i_{dofs}, \quad p = 2, 3, \ldots, N_{\text{processors}}$$

11: add offset to all local degree of freedom indices

3.5.3 Parallelizing the Adaptive Mesh Refinement Process

The adaptive mesh refinement process involves the following steps:

1. Estimate the error in the solution and select elements for refinement and coarsening.
2. Perform mesh refinement and coarsening.
3. Project data from the old mesh onto the new mesh.
4. Repartition and perform dynamic load balancing as necessary.

Parallelization of the first step requires that the error estimator be locally computable on each subdomain. This can be performed in a fashion similar to the local element computations performed in the matrix assembly phase. The flux jump error indicator in Equation (2.1), for example, is performed in parallel by computing contributions element-by-element for each subdomain.

Similarly, mesh refinement can occur in parallel with each processor performing refinement on its local elements. The major concern in this step is that any new nodes

added to the mesh are singly defined. That is, care must be taken to ensure that new nodes on processor boundaries are not added twice. This step may be performed by computing an identifying key for each node added and reconciling duplicate keys which may occur such that the topology of the refined mesh is consistent with the original mesh.

The local solution projection operators described in Section 2.4 allow for the third step to be parallelized. The required old mesh and corresponding solution data exist on each processor, and each processor will compute the projection onto the new elements (or restriction onto the coarsened mesh). It is desirable to perform this step before any repartitioning of the mesh is performed because the data required for the projection step already reside local to the processor. Therefore, in this approach the projection/restriction operators do not require inter-processor communication.

Finally, the refined mesh may no longer be well balanced across all processors. In this case some repartitioning algorithm must be applied and some number of elements (and associated data) must be migrated between processors. Note that this step has the highest communication overhead and therefore must be carefully implemented. The repartitioning algorithm must strive to rebalance the mesh while requiring as little data migration as possible. Diffusion-based algorithms are one method that can be used. They attempt to rebalance the mesh by small changes in the location of the inter-processor boundaries, but this is not done here yet.

The particular architecture of a parallel machine may need to be considered when applying a repartitioning and load balancing algorithm. In particular, on machines with very high bandwidth inter-processor interconnects it would likely be most beneficial to perform an optimal repartitioning such that the resulting mesh is well balanced. This approach would trade load balancing for (relatively inexpensive) communication requirements. Conversely, on a machine with a particularly low bandwidth interconnect the most efficient approach may be to accept additional computational load imbalance in order to avoid significant communication overhead incurred by an optimal repartitioning algorithm.

3.5.4 Input/Output Considerations

One potential bottleneck for parallel performance in any application is the input/output (I/O) process. This is particularly true on parallel clusters composed of commodity machines as the I/O subsystem performance is often not optimized for simultaneous access by parallel processes. For modest problem sizes, the mesh I/O can be serialized through one processor without suffering a substantial performance hit because the underlying storage medium is the process bottleneck.

In the current implementation, the mesh is read in its entirety in serial by processor 1 and broadcast to all the other processors. One immediate improvement to this process (which retains the serial read/write operation) is to read subsets of the mesh data structures in blocks and broadcast these blocks to all processors. By using such a buffering process a mesh which is larger than the amount of physical memory available to any single processor may be used in a simulation, as each processor in general will require only a subset of the blocks in the mesh. The current data structures could easily be generalized using this approach. Each processor would then be required only to store its "local" elements and any adjacent elements which are connected via any node to these local elements.

Note that in this approach the *entire* file containing the mesh is still read by one processor from disk in serial. Memory limitations are overcome by essentially buffering the input stream, but the scalability of this approach is limited by its serial nature. An alternative procedure can be envisioned where each processor opens the file and seeks to its relevant location directly. This approach works only in the case of a fast parallel file system which is required to avoid disk contention amongst the processors. Alternate implementations are possible in which the mesh is preprocessed and subdivided into a number of individual files which correspond to each processor. Again, note that the preprocessing step is likely to be serial.

In summary, there are a number of possibilities which exist for dealing with the scalability of I/O. Many of these approaches are necessarily serial in one aspect or another.

This reflects the serial nature of the typical mesh generation → solution generation loop. It is envisioned that in the future the proliferation of parallel mesh generation software could bypass this issue by coupling a parallel mesh generator directly to a parallel application code.

3.5.5 Performance Expectations

The parallelization strategy outlined above clearly imposes some additional overhead on certain fundamental library transactions, as is often the case in parallel algorithms. That is to say, while a well-written parallel algorithm can always be run on a single processor, the implementation may not be optimal for this degenerate non-parallel case.

For this reason it is not suggested that the current, global mesh representation be abandoned outright and replaced with the parallel strategy outlined above. Rather, the object-oriented design approach naturally supports multiple implementations of the underlying mesh technology with minimal impact on other portions of the library and no impact on external user code.

In this way a fully parallel implementation should be seen as an augmentation of current capabilities which may be used when it provides superior scalability or performance. Previous experience with the global mesh model has shown reasonable scalability to modest numbers of processors. Further, the global mesh approach vastly reduces much of the communication required during the adaptive mesh refinement process. For this reason it is expected that, on particular parallel platforms, the current approach would likely outperform the fully parallel approach on a modest number of processors. Of course, since the current implementation is not scalable, there will be a point at which the parallel implementation performance is superior. However, this crossover point is necessarily platform dependent as it depends on the configuration of a given machine. Therefore, the freedom to switch between the current global mesh approach and a fully parallel implementation should be maintained.

Summary

This chapter discusses extensions of the data structures presented in Chapter 2 to allow for their efficient implementation on parallel computers. A software implementation has been created in which a global representation of the mesh is stored on each processor and parallelization occurs at the matrix assembly, linear algebra, and error estimation levels. A strategy for parallelizing the storage of the mesh and all associated aspects of the adaptive mesh refinement process was also presented. The remaining chapters will apply the parallel adaptive technology discussed here to two distinct application classes.

Chapter 4

Biological Transport

4.1 Introduction

Although the formation of spatial patterns is a central issue in biology the mechanisms which generate them are generally poorly understood. Recent interdisciplinary collaboration has yielded mathematical models which capture the key biological processes involved in pattern formation and illustrate the unique roles played by different physical phenomenon. These models generally consist of a coupled system of nonlinear partial differential equations.

Of particular interest in the present work is how local interactions produce global patterns in bacterial populations. Such patterns have been observed experimentally by a number of researchers [16], but only recently have mathematical models been developed which provide insight into how these patterns actually develop [17]. Very complex patterns have been observed in colonies of *Escherichia coli* (*E.coli*) in a 0.24% water agar semi–solid medium [18]. Several experimentally observed patterns are presented in Figure 4.1.

The governing equations for this class of problems are of nonlinear reaction–diffusion type and will be described in the following section. These equations pose a number of challenges for accurate simulation including nonlinear coupling, rapid transients, and highly localized, transient features which must be addressed [19]. It is interesting to note that while such features make this class of problems particularly well-suited for adaptive simulation techniques, there has been very little adaptive work for biological systems. The goal of the present work is illustrate how the adaptive mesh refinement (AMR) techniques presented

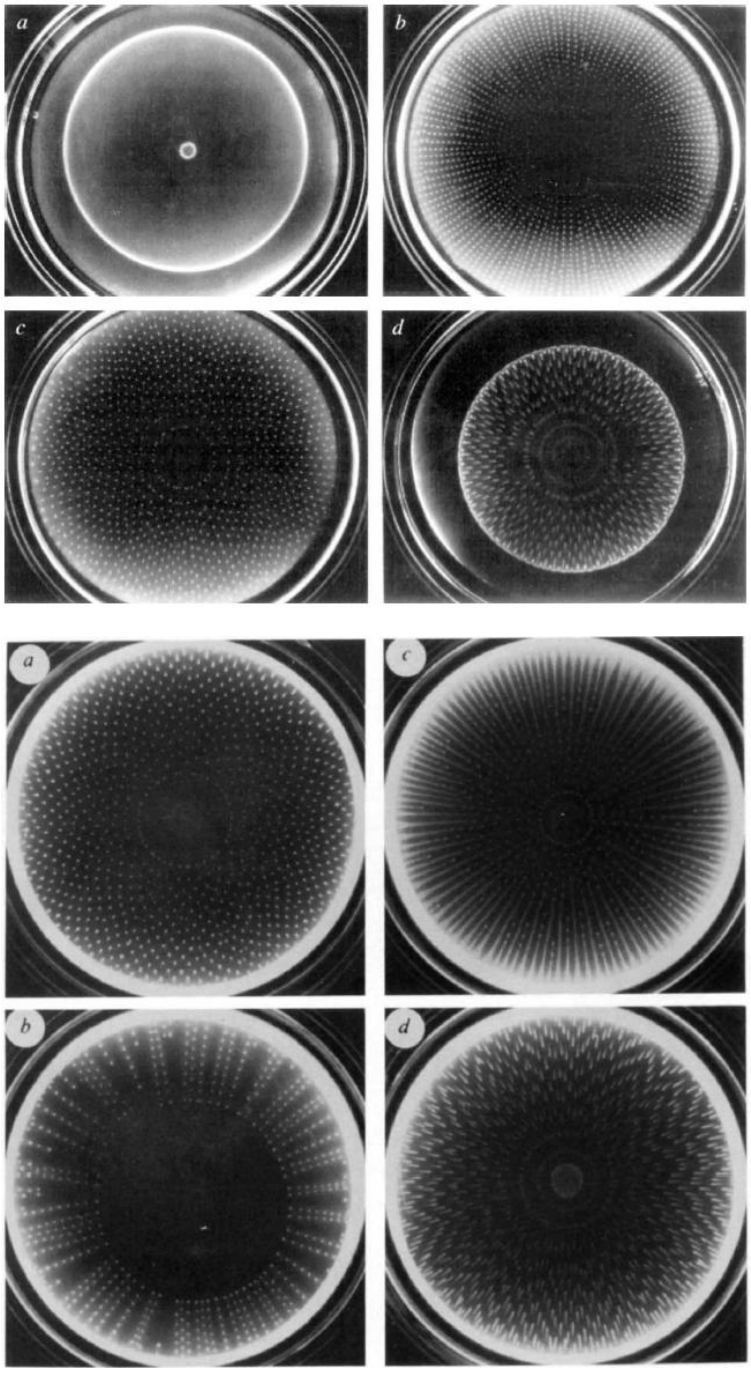


Figure 4.1: Pattern formation in *E.coli* [18],[16]

in Chapters 2 and 3 can be used efficiently in a parallel environment to address this class of problems. The use of adaptive techniques for this problem class enables high-resolution simulations of physical phenomena that would be intractable with uniform meshes.

4.2 Mathematical Model

4.2.1 A Nonlinear Reaction-Diffusion Model for Chemotactic Systems

E.coli patterns are formed when the bacteria are exposed to an external stimulant, which can be thought of as a food source. As a result of processing this stimulant the bacteria secrete aspartate, which is a strong chemoattractant. This chemoattractant, in turn, induces chemotaxis, a process which essentially attracts bacteria. The bacteria sense the ambient chemoattractant levels and move toward regions of higher concentration. When chemotaxis is sufficiently strong it competes with diffusion and spatial patterns may develop. Additionally, there are metabolic processes governing chemoattractant production and stimulant uptake which must be included in any model of such systems. Finally, the birth and death of bacteria must be modeled. This gives rise to a system for three variables: the cell density n, the chemoattractant concentration c, and the stimulant concentration s. The interplay of these physical processes is shown schematically in Figure 4.2, and their mathematical treatment will now be discussed.

One possible mathematical model for bacterial chemotaxis is given by Murray et al. [17]:

$$\frac{\partial n}{\partial t} = D_n \Delta n - \boldsymbol{\nabla} \cdot \left[\frac{k_1 n}{(k_2 + c)^2} \boldsymbol{\nabla} c \right] + k_3 n \left(k_4 \frac{s^2}{k_9 + s^2} - n \right) \tag{4.1}$$

$$\frac{\partial c}{\partial t} = D_c \Delta c + k_5 s \frac{n^2}{k_6 + n^2} - k_7 n c \tag{4.2}$$

$$\frac{\partial s}{\partial t} = D_s \Delta s - k_8 n \frac{s^2}{k_9 + s^2} \tag{4.3}$$

where n, c, and s are the concentrations of bacteria, chemoattractant, and stimulant, respectively. The remaining parameters govern the rates of proliferation, stimulant uptake,

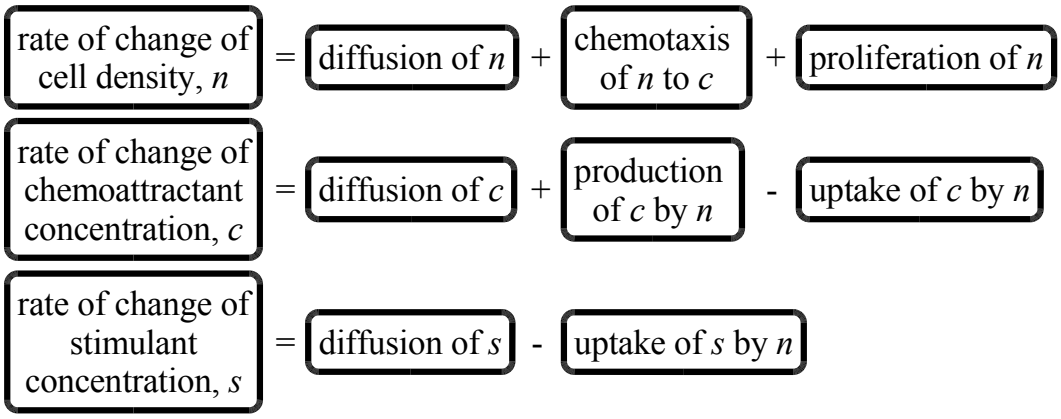


Figure 4.2: Conceptual model of bacteria/chemoattractant/stimulant system based on [17].

diffusion, and chemotaxis. Some of these parameters have been measured experimentally, while others must be estimated. It should be noted that the modeling of all right-hand-side terms in Equations (4.1)–(4.3) other than diffusion is largely phenomenological in that other functional forms for these terms are plausible.

Equation (4.1) describes the transport of bacteria. The density of bacteria, n, may change through the physical processes of diffusion, chemotaxis, and proliferation. These processes are described by the three terms on the right-hand-side of (4.1). Diffusion is the natural tendency for highly localized populations to spread out by moving in a direction away from the concentrated population (the direction opposite of the concentration gradient). Conversely, chemotaxis is a process in which bacteria are attracted to regions of high chemoattractant density. In this case the bacteria move in the direction of increasing chemoattractant (in the direction aligned with the chemoattractant gradient), hence chemotaxis may be thought of as an anti-diffusion process. Finally, the proliferation of bacteria governs the growth or decay of a population in response to the local stimulant concentration. This term may be positive, negative, or zero depending on the local density of bacteria and stimulant. In this way the proliferation term seeks to drive a given population into equilibrium with the available stimulant.

Equation (4.2) describes the transport of chemoattractant. The three terms on the right-hand-side of the equation describe transport through diffusion, production, and depletion, respectively. The first term describes the familiar process of diffusion. The second term governs the production of chemoattractant which occurs when a bacteria concentration and stimulant are in contact. This term is always positive or zero. Conversely, the third term models depletion and may be negative or zero, depending upon local conditions. This term represents the uptake of chemoattractant by bacteria.

Finally, Equation (4.3) governs the transport of stimulant. In this model the local stimulant concentration may change through either diffusion or consumption. There is no replenishing of stimulant in this model, although the degenerate case when $k_8 = 0$ corresponds to a fixed total amount of stimulant (for all time) whose spatial distribution may change only through diffusion. In this case any initial stimulant distribution will tend to diffuse until it is distributed uniformly throughout the domain.

Equations (4.1)–(4.3) can be scaled by introducing the following relationships [17]:

$$\hat{t} = k_7 n_0 t \tag{4.4}$$

$$u = \frac{n}{n_0} \tag{4.5}$$

$$v = \frac{c}{k_2} \tag{4.6}$$

$$w = \frac{s}{\sqrt{k_9}} \tag{4.7}$$

$$\hat{\Delta} = \frac{D_c}{k_7 n_0}\Delta \tag{4.8}$$

$$d_u = \frac{D_n}{D_c} \tag{4.9}$$

$$d_w = \frac{D_s}{D_c} \tag{4.10}$$

$$\alpha = \frac{k_1}{k_2 D_c} \tag{4.11}$$

$$\rho = \frac{k_3}{k_7} \tag{4.12}$$

$$\delta = \frac{k_4}{n_0} \tag{4.13}$$

$$\beta = k_5 \frac{\sqrt{k_9}}{k_7 k_2 n_0} \tag{4.14}$$

$$\kappa = \frac{k_8}{k_7 \sqrt{k_9}} \tag{4.15}$$

$$\mu = \frac{k_6}{n_0^2} \tag{4.16}$$

which produces the nondimensional system

$$\frac{\partial u}{\partial \hat{t}} = d_u \hat{\Delta} u - \alpha \boldsymbol{\nabla} \cdot \left[\frac{u}{(1+v)^2} \boldsymbol{\nabla} v \right] + \rho u \left(\delta \frac{w^2}{1+w^2} - u \right) \tag{4.17}$$

$$\frac{\partial v}{\partial \hat{t}} = \hat{\Delta} v + \beta w \frac{u^2}{\mu + u^2} - uv \tag{4.18}$$

$$\frac{\partial w}{\partial \hat{t}} = d_w \hat{\Delta} w - \kappa u \frac{w^2}{1+w^2} \tag{4.19}$$

For notational convenience the $\hat{()}$ will be dropped in the following sections.

The initial conditions for the system are constructed to imitate a quiescent domain Ω with a given stimulant concentration w_0 at time $t = t_0$. Since the domain initially is devoid of bacteria there is no initial chemoattractant concentration. An initial inoculum of bacteria, u_0, is introduced and the solution evolves in response to this 'disturbance.' Specifically, for the problems that are considered subsequently

$$u_0 = u(x, y), \quad (x, y) \in \Omega \tag{4.20}$$

$$v_0 = 0 \tag{4.21}$$

$$w_0 = w(x, y), \quad (x, y) \in \Omega \tag{4.22}$$

The actual implementation of initial conditions is an important issue. Specifically, the initial conditions must be specified and implemented in a consistent way across a range of computational meshes. This important aspect of the simulation will be addressed further in the application studies.

For a closed container with impermeable walls a physically sensible boundary condition for all variables in the system is

$$\frac{\partial u}{\partial n} = \boldsymbol{\nabla} u \cdot \hat{\boldsymbol{n}} = 0 \tag{4.23}$$

$$\frac{\partial v}{\partial n} = \boldsymbol{\nabla} v \cdot \hat{\boldsymbol{n}} = 0 \tag{4.24}$$

$$\frac{\partial w}{\partial n} = \boldsymbol{\nabla} w \cdot \hat{\boldsymbol{n}} = 0 \tag{4.25}$$

which simply states that there is no flux of bacteria, chemoattractant, or stimulant through the boundary of the domain Ω.

4.2.2 Weak Formulation

The corresponding weighted-residual statement for equations (4.17)–(4.18) follows from multiplying each equation by a suitable test function ϕ and integrating over the domain Ω, giving

$$\int_\Omega \left[\frac{\partial u}{\partial t} - d_u \Delta u + \alpha \boldsymbol{\nabla} \cdot \left[\frac{u}{(1+v)^2}\boldsymbol{\nabla} v\right] - \rho u \left(\delta \frac{w^2}{1+w^2} - u\right)\right] \phi \, d\Omega = 0 \tag{4.26}$$

$$\int_\Omega \left[\frac{\partial v}{\partial t} - \Delta v - \beta w \frac{u^2}{\mu + u^2} + uv\right] \phi \, d\Omega = 0 \tag{4.27}$$

$$\int_\Omega \left[\frac{\partial w}{\partial t} - d_w \Delta w + \kappa u \frac{w^2}{1+w^2}\right] \phi \, d\Omega = 0 \tag{4.28}$$

Applying Gauss' divergence theorem, selectively, and substituting (4.23)-(4.25) in the resulting boundary integrals yields the weak formulation: Find (u, v, w) satisfying the specified initial conditions and

$$\begin{aligned}\int_\Omega \left[\frac{\partial u}{\partial t} - \rho u \left(\delta \frac{w^2}{1+w^2} - u\right)\right] \phi \, d\Omega \\ + \int_\Omega \left[d_u \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} \phi - \alpha u \frac{\boldsymbol{\nabla} v \cdot \boldsymbol{\nabla} \phi}{(1+v)^2}\right] \, d\Omega = 0\end{aligned} \tag{4.29}$$

$$\int_\Omega \left[\left(\frac{\partial v}{\partial t} - \beta w \frac{u^2}{\mu + u^2} + uv\right)\phi + \boldsymbol{\nabla} v \cdot \boldsymbol{\nabla} \phi\right] \, d\Omega = 0 \tag{4.30}$$

$$\int_\Omega \left[\left(\frac{\partial w}{\partial t} + \kappa u \frac{w^2}{1+w^2}\right)\phi + d_w \boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} \phi\right] \, d\Omega = 0 \tag{4.31}$$

for all admissible test functions ϕ. The weak statement (4.29)-(4.31) implies (4.17)-(4.19) with (4.23)-(4.25) as natural boundary conditions.

4.2.3 Finite Element Formulation

The finite element formulation for (4.29)–(4.31) follows upon replacing (u, v, w) and ϕ with the finite dimensional approximations (u_h, v_h, w_h) and ϕ_h. Specifically, for a

standard Lagrange finite element basis we have

$$u_h(\boldsymbol{x},t) = \sum_{j=1}^{N} u_j(t)\phi_j(\boldsymbol{x}) \tag{4.32}$$

$$v_h(\boldsymbol{x},t) = \sum_{j=1}^{N} v_j(t)\phi_j(\boldsymbol{x}) \tag{4.33}$$

$$w_h(\boldsymbol{x},t) = \sum_{j=1}^{N} w_j(t)\phi_j(\boldsymbol{x}) \tag{4.34}$$

where $u_j(t)$, $v_j(t)$, and $w_j(t)$ are the nodal values of the bacteria, chemoattractant, and stimulant at time t, respectively, and N is the number of nodes in the domain. The corresponding discrete weak statement is then: find the approximate solution (u_h, v_h, w_h) satisfying the specified initial conditions and

$$\int_\Omega \left[\frac{\partial u_h}{\partial t} - \rho u_h \left(\delta \frac{w_h^2}{1+w_h^2} - u_h \right) \right] \phi_h \, d\Omega + \int_\Omega \left[d_u \boldsymbol{\nabla} u_h \cdot \boldsymbol{\nabla}\phi_h - \alpha u_h \frac{\boldsymbol{\nabla} v_h \cdot \boldsymbol{\nabla}\phi_h}{(1+v_h)^2} \right] d\Omega = 0 \tag{4.35}$$

$$\int_\Omega \left[\left(\frac{\partial v_h}{\partial t} - \beta w_h \frac{u_h^2}{\mu + u_h^2} + u_h v_h \right) \phi_h + \boldsymbol{\nabla} v_h \cdot \boldsymbol{\nabla}\phi_h \right] d\Omega = 0 \tag{4.36}$$

$$\int_\Omega \left[\left(\frac{\partial w_h}{\partial t} + \kappa u_h \frac{w_h^2}{1+w_h^2} \right) \phi + d_w \boldsymbol{\nabla} w_h \cdot \boldsymbol{\nabla}\phi_h \right] d\Omega = 0 \tag{4.37}$$

for all admissible test functions ϕ_h. This discrete approximation to (4.29)-(4.31) forms the basis of the computational model described in the next section.

4.3 Solution Methodology

Equations (4.17)–(4.19) form a highly coupled, transient, nonlinear system. Furthermore, the system evolution occurs over a disparate range of time scales that must be captured accurately. This section outlines the techniques employed in the present work to solve the system of equations.

4.3.1 Time Integration

The time integration method used in this work can be developed by considering the generic first-order ordinary differential equation

$$\dot{\boldsymbol{u}} = \boldsymbol{f}\left(t, \boldsymbol{u}\left(t\right)\right), \quad t > t_0, \quad \boldsymbol{u}\left(t_0\right) = \boldsymbol{u}_0 \tag{4.38}$$

Explicit methods for (4.38) are simple to formulate and the computational cost per step is low. The price for this simplicity is decreased stability. Consequently, limits must be posed on the integration step size Δt in an explicit scheme. Implicit methods, on the other hand, are typically stable for any Δt but are considerably more difficult to implement and have a higher computational cost per time step. A combined explicit Adams-Bashforth predictor and implicit Trapezoidal corrector with step control are applied in the present work and are discussed next.

Adams-Bashforth Explicit Scheme

The Adams–Bashforth explicit integration formula applied to Equation (4.38) is given by

$$\boldsymbol{u}\left(t + \Delta t\right) = \boldsymbol{u}_n + \frac{\Delta t}{2}\left[\left(2 + \frac{\Delta t_n}{\Delta t_{n-1}}\right)\boldsymbol{f}_n - \left(\frac{\Delta t_n}{\Delta t_{n-1}}\right)\boldsymbol{f}_{n-1}\right] + \mathcal{O}\left(\Delta t^2\right) \tag{4.39}$$

where $\boldsymbol{u}_n = \boldsymbol{u}\left(t_n\right)$, $\Delta t_n = t_n - t_{n-1}$, and $\boldsymbol{f}_n = \boldsymbol{f}\left(t_n, \boldsymbol{u}\left(t_n\right)\right)$. This can be used to give a second–order accurate approximation for $\boldsymbol{u}_{n+1}$ by ignoring the error terms $\mathcal{O}\left(\Delta t^2\right)$

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \frac{\Delta t_{n+1}}{2}\left[\left(2 + \frac{\Delta t_n}{\Delta t_{n-1}}\right)\boldsymbol{f}_n - \left(\frac{\Delta t_n}{\Delta t_{n-1}}\right)\boldsymbol{f}_{n-1}\right] \tag{4.40}$$

Substituting Equation (4.38) into Equation (4.40) gives the equivalent form

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \frac{\Delta t_{n+1}}{2}\left[\left(2 + \frac{\Delta t_n}{\Delta t_{n-1}}\right)\dot{\boldsymbol{u}}_n - \left(\frac{\Delta t_n}{\Delta t_{n-1}}\right)\dot{\boldsymbol{u}}_{n-1}\right] \tag{4.41}$$

Trapezoidal Rule Implicit Scheme

The implicit trapezoidal integration rule applied to Equation (4.38) is given by

$$\boldsymbol{u}\left(t + \Delta t\right) = \boldsymbol{u}_n + \frac{\Delta t}{2}\left(\boldsymbol{f}\left(t + \Delta t\right) + \boldsymbol{f}_n\right) + \mathcal{O}\left(\Delta t^2\right) \tag{4.42}$$

which yields the second order accurate scheme for $\boldsymbol{u}_{n+1}$

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \frac{\Delta t_{n+1}}{2}\left(\boldsymbol{f}_{n+1} + \boldsymbol{f}_n\right) \tag{4.43}$$

or equivalently

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \frac{\Delta t_{n+1}}{2}\left(\dot{\boldsymbol{u}}_{n+1} + \dot{\boldsymbol{u}}_n\right) \tag{4.44}$$

This scheme is implicit because of its dependence on the unknown value $\boldsymbol{f}_{n+1}$. Furthermore, for the class of problems considered in this work $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{u})$ and is highly nonlinear. Therefore, using Equation (4.43) requires the solution of a nonlinear implicit system of equations using the techniques described in Section 4.3.2.

Local Error Control

An important benefit of the two stage predictor–corrector algorithm is that it provides a means to estimate the local error incurred at a given time step [20]. Consider $\hat{\boldsymbol{u}}_{n+1}$ and $\boldsymbol{u}_{n+1}$ as predictor and corrector calculations, respectively, both of which approximate the exact solution $\boldsymbol{u}\left(t_{n+1}\right)$ to $\mathcal{O}(p)$ accuracy. The truncation errors for these approximate solutions are defined as

$$\hat{\boldsymbol{\tau}}_{n+1} \equiv \boldsymbol{u}\left(t_{n+1}\right) - \hat{\boldsymbol{u}}_{n+1} = \hat{c}\Delta t_{n+1}^{p+1}\boldsymbol{u}^{(p+1)}\left(t_{n+1}\right) + \mathcal{O}\left(\Delta t_{n+1}^{p+2}\right) \tag{4.45}$$

$$\boldsymbol{\tau}_{n+1} \equiv \boldsymbol{u}\left(t_{n+1}\right) - \boldsymbol{u}_{n+1} = c\Delta t_{n+1}^{p+1}\boldsymbol{u}^{(p+1)}\left(t_{n+1}\right) + \mathcal{O}\left(\Delta t_{n+1}^{p+2}\right) \tag{4.46}$$

Subtracting (4.45) from (4.46) and ignoring higher order terms yields

$$\begin{aligned} \hat{\boldsymbol{u}}_{n+1} - \boldsymbol{u}_{n+1} &\approx \left(c - \hat{c}\right)\Delta t_{n+1}^{p+1}\boldsymbol{u}^{(p+1)}\left(t_{n+1}\right) \\ \Delta t_{n+1}^{p+1}\boldsymbol{u}^{(p+1)}\left(t_{n+1}\right) &\approx \frac{\hat{\boldsymbol{u}}_{n+1} - \boldsymbol{u}_{n+1}}{c - \hat{c}} \end{aligned} \tag{4.47}$$

Substituting (4.47) into (4.46) provides an estimation of the local truncation error $\boldsymbol{\tau}_{n+1}$

$$\boldsymbol{\tau}_{n+1} \approx \frac{c}{c - \hat{c}}\left(\hat{\boldsymbol{u}}_{n+1} - \boldsymbol{u}_{n+1}\right) \tag{4.48}$$

which, for the Adams–Bashforth predictor/trapezoidal rule corrector pair employed in the present work becomes

$$\boldsymbol{\tau}_{n+1} \approx \frac{\hat{\boldsymbol{u}}_{n+1} - \boldsymbol{u}_{n+1}}{3\left(1 + \Delta t_n / \Delta t_{n+1}\right)} \tag{4.49}$$

Finally, Equation (4.46) yields the following relationship for the truncation error at successive time steps

$$\frac{\|\boldsymbol{\tau}_{n+2}\|}{\|\boldsymbol{\tau}_{n+1}\|} \approx \left(\frac{\Delta t_{n+2}}{\Delta t_{n+1}}\right)^3 \tag{4.50}$$

which can be used to provide an estimate for the time step Δt_{n+2} required to limit $\|\boldsymbol{\tau}_{n+2}\|$ to some specified value ε_t

$$\Delta t_{n+2} = \Delta t_{n+1} \left(\frac{\varepsilon_t}{\|\boldsymbol{\tau}_{n+1}\|}\right)^{1/3} \tag{4.51}$$

Predictor–Corrector Algorithm

The general approach is to use the explicit Adams–Bashforth step given by Equation (4.41) to predict $\hat{\boldsymbol{u}}_{n+1}$. This predicted value is then used as the initial iterate to solve the nonlinear system in Equation (4.43) via Newton iteration (as described in the next section). The algorithm is outlined as follows:

1. Predict the solution $\hat{\boldsymbol{u}}_{n+1}$ at t_{n+1} using Equation (4.41):

$$\hat{\boldsymbol{u}}_{n+1} = \boldsymbol{u}_n + \frac{\Delta t_{n+1}}{2}\left[\left(2 + \frac{\Delta t_n}{\Delta t_{n-1}}\right)\dot{\boldsymbol{u}}_n - \left(\frac{\Delta t_n}{\Delta t_{n-1}}\right)\dot{\boldsymbol{u}}_{n-1}\right]$$

2. Solve Equation (4.43) using the predicted solution $\hat{\boldsymbol{u}}_{n+1}$ as the initial guess for the nonlinear implicit solver.

3. Estimate the truncation error $\|\boldsymbol{\tau}_{n+1}\|$ using (4.49) and compute the next time step Δt_{n+2} using (4.51):

$$\|\boldsymbol{\tau}_{n+1}\| = \frac{\|\hat{\boldsymbol{u}}_{n+1} - \boldsymbol{u}_{n+1}\|}{3\left(1 + \Delta t_n / \Delta t_{n+1}\right)}$$

$$\Delta t_{n+2} = \Delta t_{n+1}\left(\frac{\varepsilon_t}{\|\boldsymbol{\tau}_{n+1}\|}\right)^{1/3}$$

4. Update the time derivative by inverting Equation (4.44)

$$\dot{\boldsymbol{u}}_{n+1} = \frac{2}{\Delta t_{n+1}} \left(\boldsymbol{u}_{n+1} - \boldsymbol{u}_n \right) - \dot{\boldsymbol{u}}_n$$

5. Terminate the integration when the change between successive time steps is less than some specified tolerance ε_{ss}

$$\| \boldsymbol{u}_{n+1} - \boldsymbol{u_n} \| < \varepsilon_{ss} \| \boldsymbol{u}_{n+1} \| \tag{4.52}$$

4.3.2 Linearization

This section addresses the solution of the nonlinear system of equations which results from the implicit time discretization. To develop the nonlinear solution scheme it is useful to recast Equation (4.43) in residual form as

$$\boldsymbol{R} \left(\boldsymbol{u}_{n+1} \right) \equiv \frac{\boldsymbol{u}_{n+1} - \boldsymbol{u}_n}{\Delta t_{n+1}} - \frac{\boldsymbol{f} \left(\boldsymbol{u}_{n+1} \right) + \boldsymbol{f} \left(\boldsymbol{u}_n \right)}{2} = 0 \tag{4.53}$$

where $\boldsymbol{u}_{n+1}$ is the unknown solution at time $t_{n+1} s$. Newton's method can be derived for this system by considering a Taylor series applied to Equation (4.53)

$$\boldsymbol{R} \left(\boldsymbol{u_{n+1}} + \delta \boldsymbol{u}_{n+1} \right) = \boldsymbol{R} \left(\boldsymbol{u}_{n+1} \right) + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}} \Big|_{\boldsymbol{u}_{n+1}} \delta \boldsymbol{u}_{n+1} + \mathcal{O} \left(\delta \boldsymbol{u}_{n+1}^2 \right) \tag{4.54}$$

where $\partial \boldsymbol{R} / \partial \boldsymbol{u}$ is the Jacobian matrix for the system. Ignoring higher order terms and requiring $\boldsymbol{R} \left(\boldsymbol{u}_{n+1} + \delta \boldsymbol{u}_{n+1} \right) = 0$ produces Newton's method

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}} \Big|_{\boldsymbol{u}_{n+1}^l} \delta \boldsymbol{u}_{n+1}^{l+1} = -\boldsymbol{R} \left(\boldsymbol{u}_{n+1}^l \right) \tag{4.55}$$

or equivalently

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}} \Big|_{\boldsymbol{u}_{n+1}^l} \boldsymbol{u}_{n+1}^{l+1} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}} \Big|_{\boldsymbol{u}_{n+1}^l} \boldsymbol{u}_{n+1}^{l} - \boldsymbol{R} \left(\boldsymbol{u}_{n+1}^l \right) \tag{4.56}$$

where $\boldsymbol{u}_{n+1}^l$ is an intermediate iterate which approximates the unknown root $\boldsymbol{u}_{n+1}$. The nonlinear system is then solved as a sequence of linear approximations as follows:

1. Let $l = 0, \ \boldsymbol{u}_{n+1}^l = \hat{\boldsymbol{u}}_{n+1}$

2. Solve the linear system (4.56) for $\boldsymbol{u}_{n+1}^{l+1}$
3. Stop if $\|\boldsymbol{u}_{n+1}^{l+1} - \boldsymbol{u}_{n+1}^{l}\| < \varepsilon_{nl}$
4. Else increment l and repeat from step 2

Newton's method exhibits quadratic convergence provided that the initial iterate $\boldsymbol{u}_{n+1}^{0}$ is sufficiently close to the root $\boldsymbol{u}_{n+1}$. If this condition is not met then the iteration may converge at a sub-optimal rate or fail to converge altogether. In this work the Adams–Bashforth predicted solution is taken as the initial guess and nonlinear convergence is typically obtained in three or four iterations. If for some reason the nonlinear scheme fails to converge within a specified number of iterations the time step is halved and the process is repeated. This successive step–halving should ensure that the predicted solution is eventually close enough to the unknown root for the method to converge.

4.3.3 Adaptive Mesh Refinement

As mentioned in the introduction, the presence of highly localized features such as propagating "fronts" and isolated, stationary "spots" makes this application class particularly well-suited to simulation with adaptive mesh refinement techniques. This section provides an overview of the algorithm used to automatically adapt the mesh to a particular solution.

The AMR algorithm requires a candidate solution on a particular mesh as input. The error in this candidate solution is then estimated in some way. The algorithm will then selectively coarsen and refine the mesh in areas of relatively low and high error, respectively. The end goal is to produce a mesh which equidistributes the error. Note that the current software implementation can only coarsen cells which have previously been refined, so it is not possible to coarsen the mesh below its initial resolution.

The gradient-based error indicator described in Section 2.5 is used here to select which elements will be coarsened and refined at each step in the adaptive process. The

gradient-based indicator is very effective at locating regions of high curvature in the solution field. This indicator is applied to all variables in the system, resulting in a discrete value for each element in the simulation.

The mean and standard deviation of this distribution is then calculated. User-specified refinement and coarsening fractions of the standard deviation are then added and subtracted from the mean, respectively, to find threshold values for coarsening and refinement. Any element whose error is less than the coarsening threshold is considered a candidate for coarsening. All elements whose error is greater than the refinement threshold will be refined, provided they do not exceed a user-specified maximum refinement level.

4.3.4 Solution Algorithm

The numerical methods described in the preceding sections are combined into the solution algorithm outlined in Algorithm 4.1. This is the basic solution procedure that is applied in the following section to a number of application studies.

Algorithm 4.1 Transient adaptive nonlinear solution algorithm used for chemotactic *E.coli* systems

Interpolate initial conditions
$\boldsymbol{u}^0 = \boldsymbol{u}(\boldsymbol{x}, t)$
for $n = 1$ to $N_{\text{time steps}}$ **do**
  Predict $\hat{\boldsymbol{u}}^n$ using (4.41)
  Let $\tilde{\boldsymbol{u}}^n = \hat{\boldsymbol{u}}^n$
  Solve the nonlinear system for $\tilde{\boldsymbol{u}}^n$:
  do
    Form system matrix $\boldsymbol{K} = \boldsymbol{K}(\tilde{\boldsymbol{u}}^n, \boldsymbol{u}^{n-1})$
    Form system vector $\boldsymbol{f} = \boldsymbol{f}(\tilde{\boldsymbol{u}}^n, \boldsymbol{u}^{n-1})$
    Solve the linear system $\boldsymbol{K}\ \delta\tilde{\boldsymbol{u}}^n = \boldsymbol{f}$
    Update the solution $\tilde{\boldsymbol{u}}^n \leftarrow \tilde{\boldsymbol{u}}^n + \delta\tilde{\boldsymbol{u}}^n$
  while $\|\delta\tilde{\boldsymbol{u}}^n\|_\infty < \varepsilon_{nl}$
  Compute error indicator for each element using $\tilde{\boldsymbol{u}}^n$
  if error is acceptable
    Let $\boldsymbol{u}^n = \tilde{\boldsymbol{u}}^n$
  else
    Refine and coarsen mesh
    Project $\Pi\tilde{\boldsymbol{u}}^n \rightarrow \boldsymbol{u}^n,\ \Pi\tilde{\boldsymbol{u}}^{n-1} \rightarrow \boldsymbol{u}^{n-1}$
    Solve the nonlinear system for $\boldsymbol{u}^n$:
    do
      Form system matrix $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{u}^n, \boldsymbol{u}^{n-1})$
      Form system vector $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{u}^n, \boldsymbol{u}^{n-1})$
      Solve the linear system $\boldsymbol{K}\ \delta\boldsymbol{u}^n = \boldsymbol{f}$
      Update the solution $\boldsymbol{u}^n \leftarrow \boldsymbol{u}^n + \delta\boldsymbol{u}^n$
    while $\|\delta\boldsymbol{u}^n\|_\infty < \varepsilon_{nl}$
  endif
end

4.4 Application Studies

4.4.1 Continuous Concentric Advancing Rings

4.4.1.1 Domain Specification and Initial Conditions

The first biological transport application investigated here corresponds to the case of relatively weak chemotaxis in the presence of strong diffusion. The initial conditions for this problem physically correspond to an initial inoculum of bacteria located at the center of a square domain defined as $\Omega = [-20, 20]^2$. The domain is initially devoid of chemoattractant and has a uniform stimulant distribution. By assuming symmetry of the spatial pattern with respect to the x and y axes it is possible to simulate only the quarter-domain $\hat{\Omega} = [0, 20]^2$, thus reducing the number of unknowns in the approximate solution by a factor of four.

In Equation (4.5) the physical bacteria concentration (n) is normalized by some reference concentration (n_0). The resulting initial concentration for this case is $u_0 = 5$. It is important that u_0 be specified in such a way that it can be implemented consistently across a range of mesh resolutions and element types so that mesh refinement studies may be performed with consistent initial data. This is especially true for reactive systems such as (4.17)–(4.19) which are extremely sensitive to initial values in the sense that slightly perturbed initial conditions can evolve into markedly different states [2].

The present work uses a sequence of uniformly refined meshes with both bilinear and biquadratic elements to assess mesh convergence. The coarsest mesh contains 100×100 elements, yielding a coarse mesh spacing of $h_c = 0.2$. For this family of meshes the initial conditions are taken as a tensor product of the following one-dimensional distributions, which may be represented exactly on all meshes used in this study:

$$u_0 = 5 \qquad x \leq h_c \tag{4.57}$$

$$= 10 - \frac{5x}{h_c} \qquad h_c < x < 2hc \tag{4.58}$$

$$= 0 \qquad x \geq 2hc \tag{4.59}$$

$$v_0 = 0 \tag{4.60}$$

$$w_0 = 5 \tag{4.61}$$

The resulting two-dimensional initial bacteria concentration is shown in Figure 4.3. In

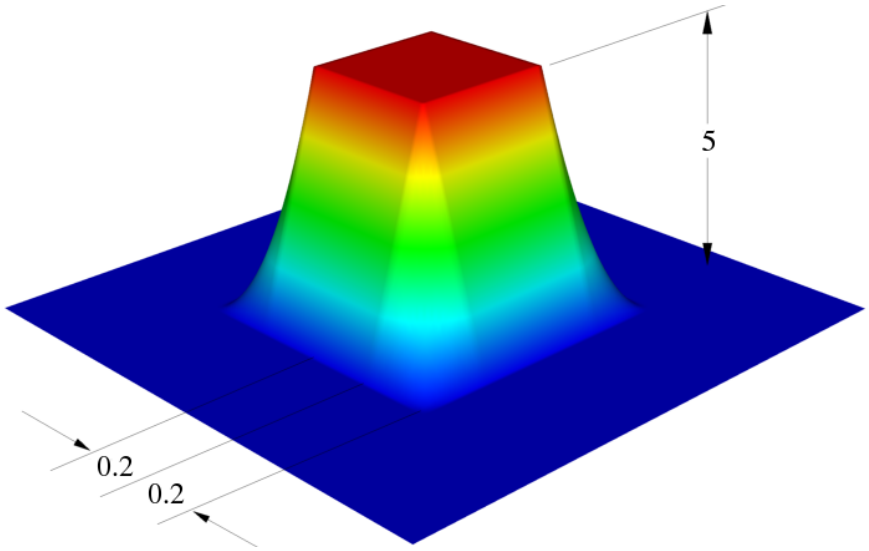


Figure 4.3: Initial bacteria concentration for the concentric ring problem

particular, the linear decay of initial bacteria concentration specified by Equation (4.58) is chosen because it can be represented exactly with the bilinear and biquadratic elements for the full range of mesh resolutions used in this study. Also, by fixing the slope of the decay for all mesh resolutions, initial transients caused by inconsistent initial conditions

are avoided. Numerical experiments have shown that failing to maintain consistent initial conditions for reactive transport applications can yield markedly different dynamics in the system.

4.4.1.2 System Parameters

The relevant parameters for this case were taken from Woodward et al. [21] and are listed in Table 4.1. The particular choice $\alpha = 7$ de-emphasizes the influence of chemo-

Table 4.1: Nondimensional parameter values for concentric rings (from Woodward et al. [21])

d_u	d_w	α	β	δ	ρ	μ	κ
1/4	1	7	10	10	1/10	250	0

taxis in comparison to problems considered subsequently. Put another way, this problem is more sensitive to transport via reaction and diffusion than chemotaxis. For this choice of parameters equations (4.17)–(4.19) become

$$\frac{\partial u}{\partial t} = \frac{1}{4}\Delta u - 7\,\boldsymbol{\nabla}\cdot\left[\frac{u}{(1+v)^2}\boldsymbol{\nabla} v\right] + \frac{u}{10}\left(\frac{10\,w^2}{1+w^2} - u\right) \tag{4.62}$$

$$\frac{\partial v}{\partial t} = \Delta v + \frac{10\,wu^2}{250+u^2} - uv \tag{4.63}$$

$$\frac{\partial w}{\partial t} = \Delta w \tag{4.64}$$

In particular, the parameter $\kappa = 0$ decouples the stimulant concentration (w) from the bacterial (u) and chemoattractant (v) concentrations. The physical interpretation of this scenario is that the stimulant supply is essentially unlimited, and its distribution is influenced only though diffusion. Specifically, the bacteria cannot deplete the stimulant source, no matter how densely populated a region may be. Additionally, since the stimulant is uniformly distributed initially, even diffusion will be absent for this particular case. In this scenario the stimulant distribution is completely defined by its initial value, and the

governing equations reduce to

$$w = 5 \tag{4.65}$$

$$\frac{\partial u}{\partial t} = \frac{1}{4}\Delta u - 7\,\boldsymbol{\nabla}\cdot\left[\frac{u}{(1+v)^2}\boldsymbol{\nabla}v\right] + u\left(\frac{25}{26} - \frac{u}{10}\right) \tag{4.66}$$

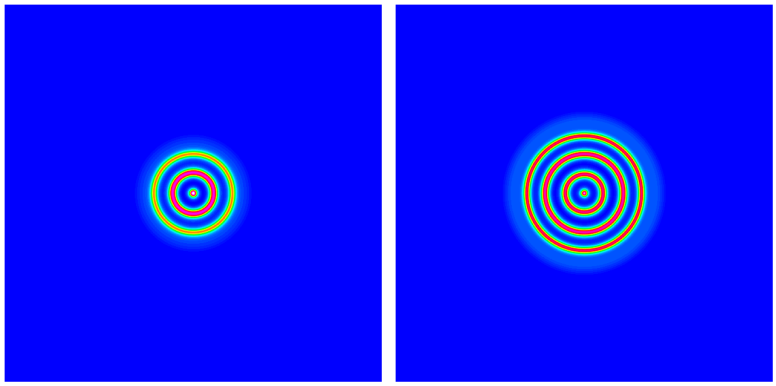
$$\frac{\partial v}{\partial t} = \Delta v + \frac{50\,u^2}{250+u^2} - uv \tag{4.67}$$

It is worth noting that the decoupling of the governing equations when $\kappa = 0$ could be exploited numerically. In this case the system Jacobian required in the nonlinear solution scheme is smaller than the general case because $\partial R_u/\partial w$, $\partial R_v/\partial w$, $\partial R_w/\partial u$, and $\partial R_w/\partial v$ are identically zero, hence no storage needs to be allocated for these terms.

4.4.1.3 Results

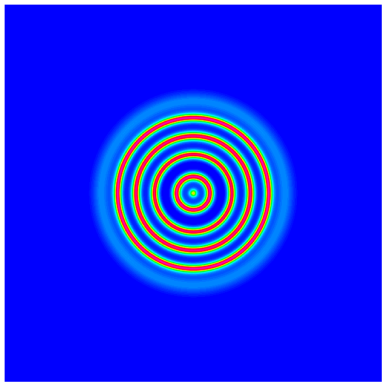
For the parameter set considered in this problem (see Table 4.1), the system forms a pattern of concentric rings which grow in number as a function of time until the domain is filled. Figure 4.4 shows a time sequence of bacterial concentration. The chemoattractant concentration for the same points in time is shown in Figure 4.5. This concentric ring pattern is similar to the top left experimental image shown in Figure 4.1.

It is interesting to note that the spatial pattern shown in the figures exhibits radial symmetry. It is not immediately obvious that this would be the case, especially given the clearly two-dimensional nature of the initial conditions (c.f. Figure 4.3). Figure 4.6 details the initial transients for the first nine timesteps in the simulation with a fixed step size $\Delta t = 2 \times 10^{-3}$. It is clear from the figure that the initial, tensor-product profile is quickly diffused. The corner regions of maximum gradient are smoothed quickly and then the entire profile begins to expand and becomes symmetric. After nine timesteps the asymmetry of the initial conditions is virtually eliminated. Based on this behavior, it is clear that the initial transients are diffusion dominated. The isotropic diffusion essentially smooths the initial conditions into a radially symmetric profile before significant reaction and chemotaxis begin. It should be possible to exploit this radial symmetry for a significant

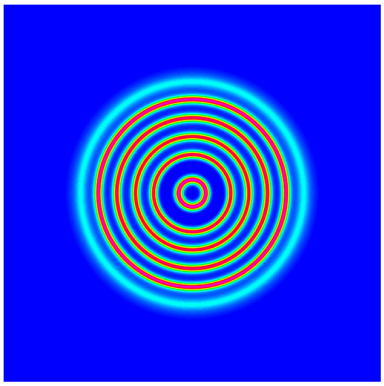


(a) t=15.0

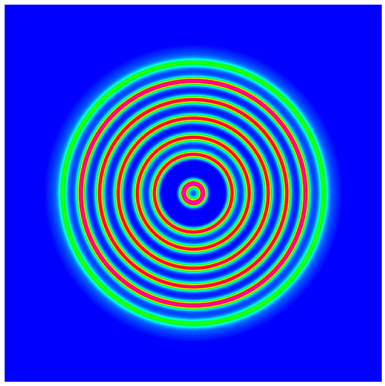
(b) t=17.5



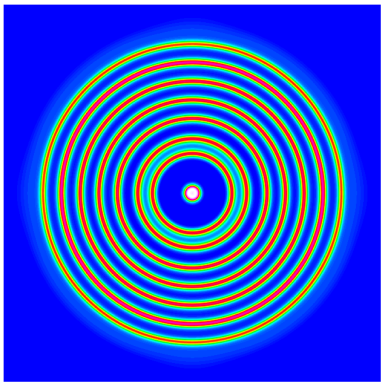
(c) t=20.0



(d) t=22.5

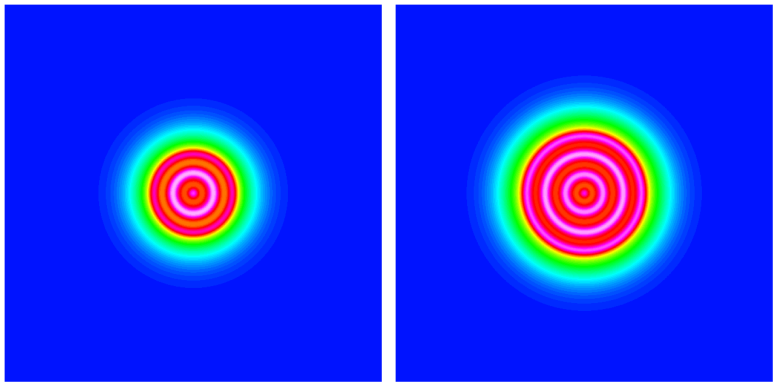


(e) t=25.0



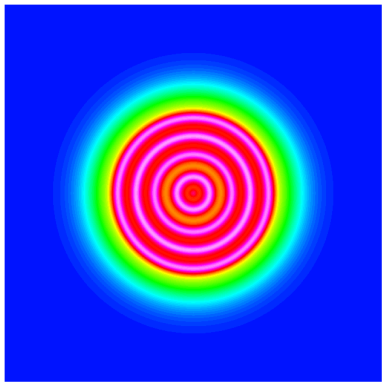
(f) t=27.5

Figure 4.4: Continuous concentric advancing rings. Bacteria concentration history.

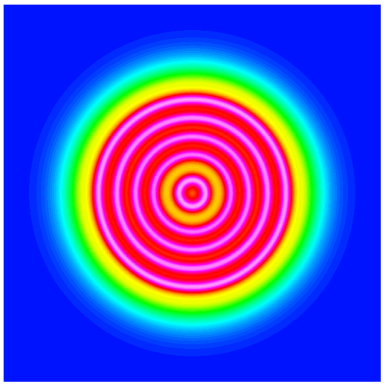


(a) t=15.0

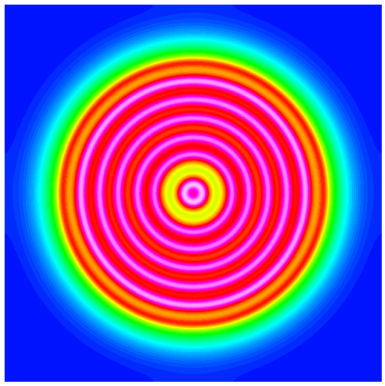
(b) t=17.5



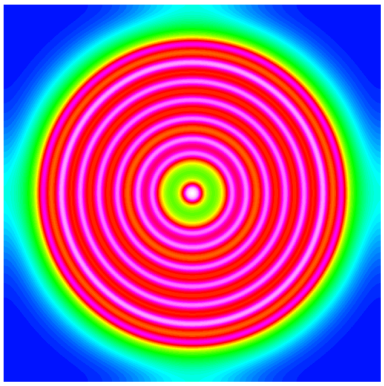
(c) t=20.0



(d) t=22.5

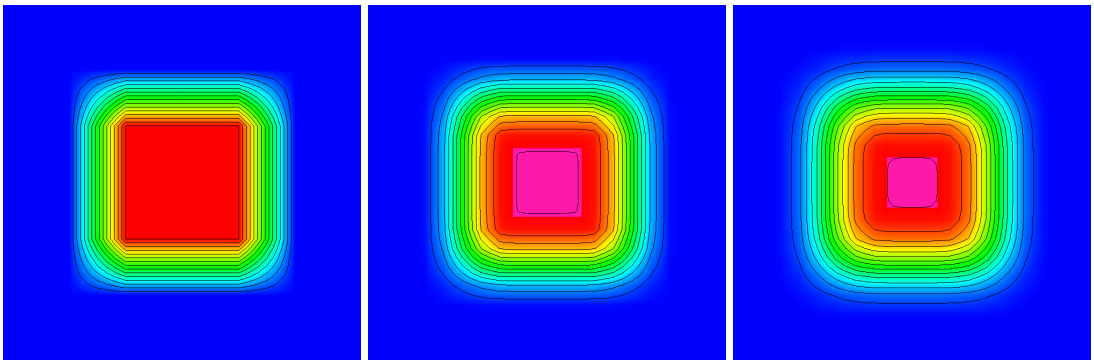


(e) t=25.0

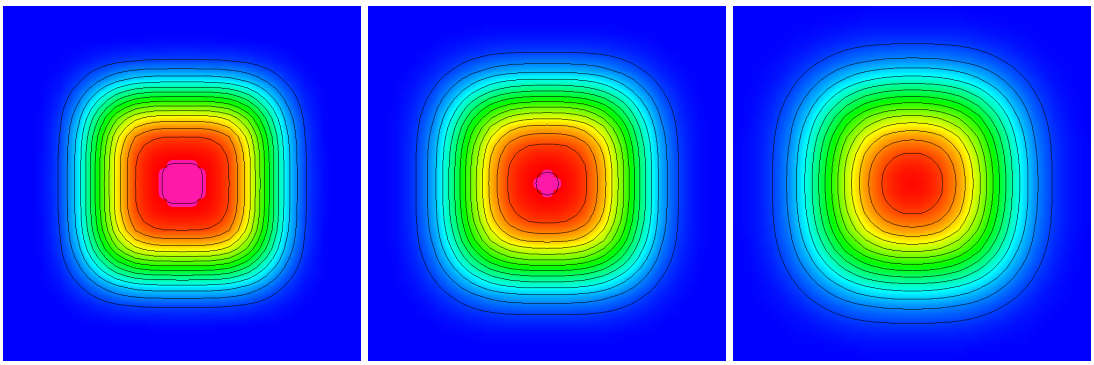


(f) t=27.5

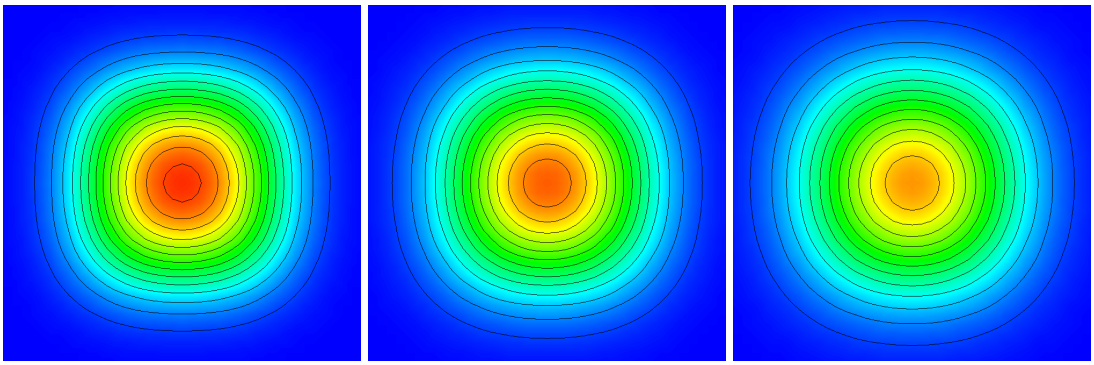
Figure 4.5: Continuous concentric advancing rings. Chemoattractant concentration history.



(a) t=0.000 (b) t=0.002 (c) t=0.004



(d) t=0.006 (e) t=0.008 (f) t=0.010



(g) t=0.006 (h) t=0.008 (i) t=0.010

Figure 4.6: Initial transient and smoothing of the initial bacteria concentration. A $[-1, 1]^2$ subregion of the mesh is shown with linear contours ranging from $[0, 5]$.

computational savings. By reformulating (4.17)–(4.19) in terms of polar (r, θ) coordinates and assuming θ-symmetry the spatial dimensionality of the system is reduced. Thus it should be possible to solve a reduced one-dimensional problem for the subset of parameters which yield radially symmetric patterns.

The maximum nondimensional bacteria and chemoattractant concentrations for this parameter set are plotted as a function of time for a series of hierarchically refined meshes in Figure 4.7. It is evident from the figure that the dynamics captured with the 100×100

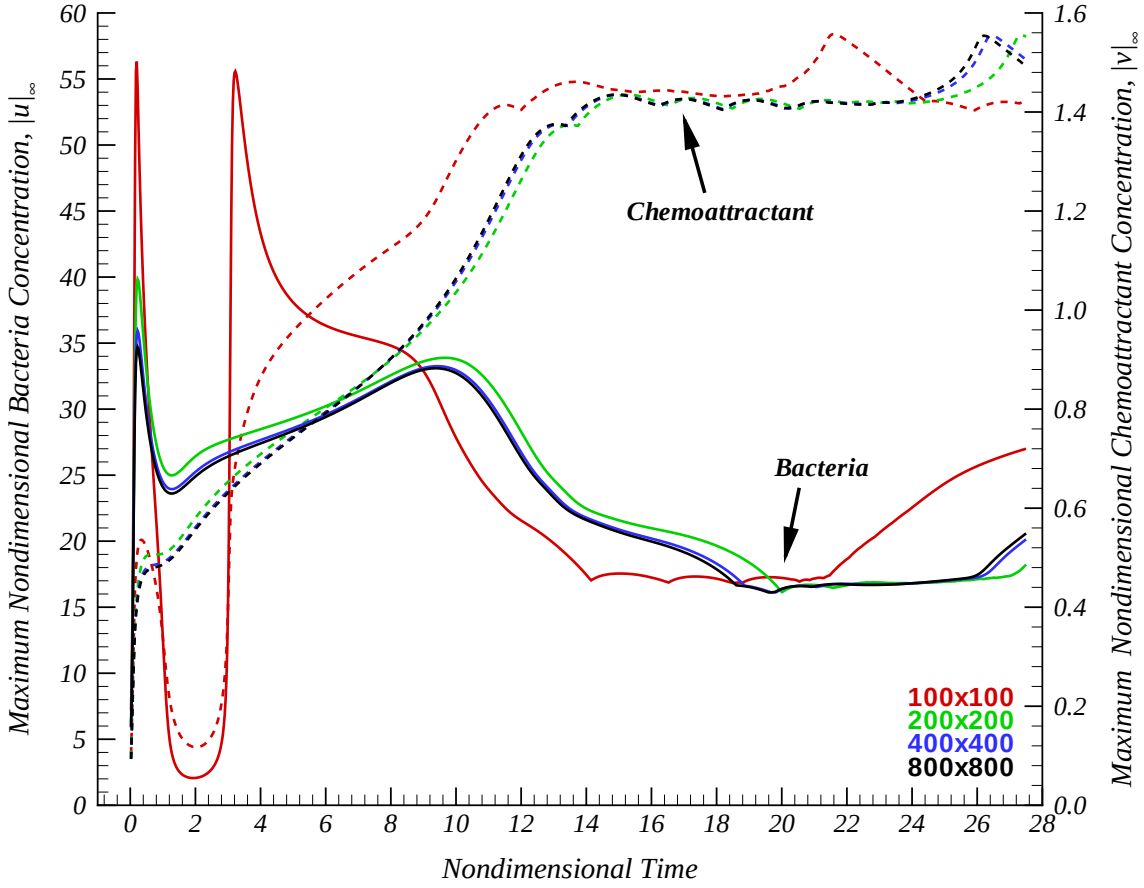


Figure 4.7: Continuous concentric advancing rings. Maximum bacteria and chemoattractant history for a sequence of meshes composed of biquadratic finite elements.

mesh are very inaccurate, while those of the 200×200, 400×400, and 800×800 meshes

essentially agree. All cases predict a sharp initial increase in bacteria concentration in the first (nondimensional) second of the simulation. Focusing now on the finer mesh results, the maximum bacteria concentration then drops to a value of $\|u\|_\infty \approx 25$ at around $t = 1$ while the maximum chemoattractant concentration steadily increases. Both then increase until $t \approx 10$. For this entire time range the system contains only a single "ring" located at the origin of the domain. That is, all the dynamics prior to $t \approx 10$ are directly tied to the initial conditions.

At subsequent times rings begin to form. During this period the maximum value of the chemoattractant is essentially constant at ≈ 1.4. Small amplitude oscillations are clearly evident which correspond to the formation of new rings. The increase in both $\|u\|_\infty$ and $\|v\|_\infty$ at the end of the simulation is attributed to interactions with the boundary of the domain. It can be seen in Figure 4.5(f) that in the final portion of the simulation the boundary is indeed affecting the solution, as expected.

Formally, the error in the solution is defined as

$$e_u = u_h - u \tag{4.68}$$

$$e_v = v_h - v \tag{4.69}$$

$$e_w = w_h - w \tag{4.70}$$

where $()_h$ denotes the approximate solution obtained on a mesh with a characteristic spacing of h and (u, v, w) are the (unknown) exact solution values. Clearly the exact solution is not known in general. However, it is possible to approximate (4.68)–(4.70) by comparing the solution on two different meshes, one being very fine. That is,

$$e_u \approx e_u^c = u_c - u_f \tag{4.71}$$

$$e_v \approx e_v^c = v_c - v_f \tag{4.72}$$

$$e_w \approx e_w^c = w_c - w_f \tag{4.73}$$

where $()_c$ and $()_f$ denote coarse and fine mesh solution values, respectively. Figures 4.8

and 4.9 use this method to illustrate the error in the 100×100, 200×200, and 400×400 biquadratic finite element meshes via comparison with the 800×800 solution.

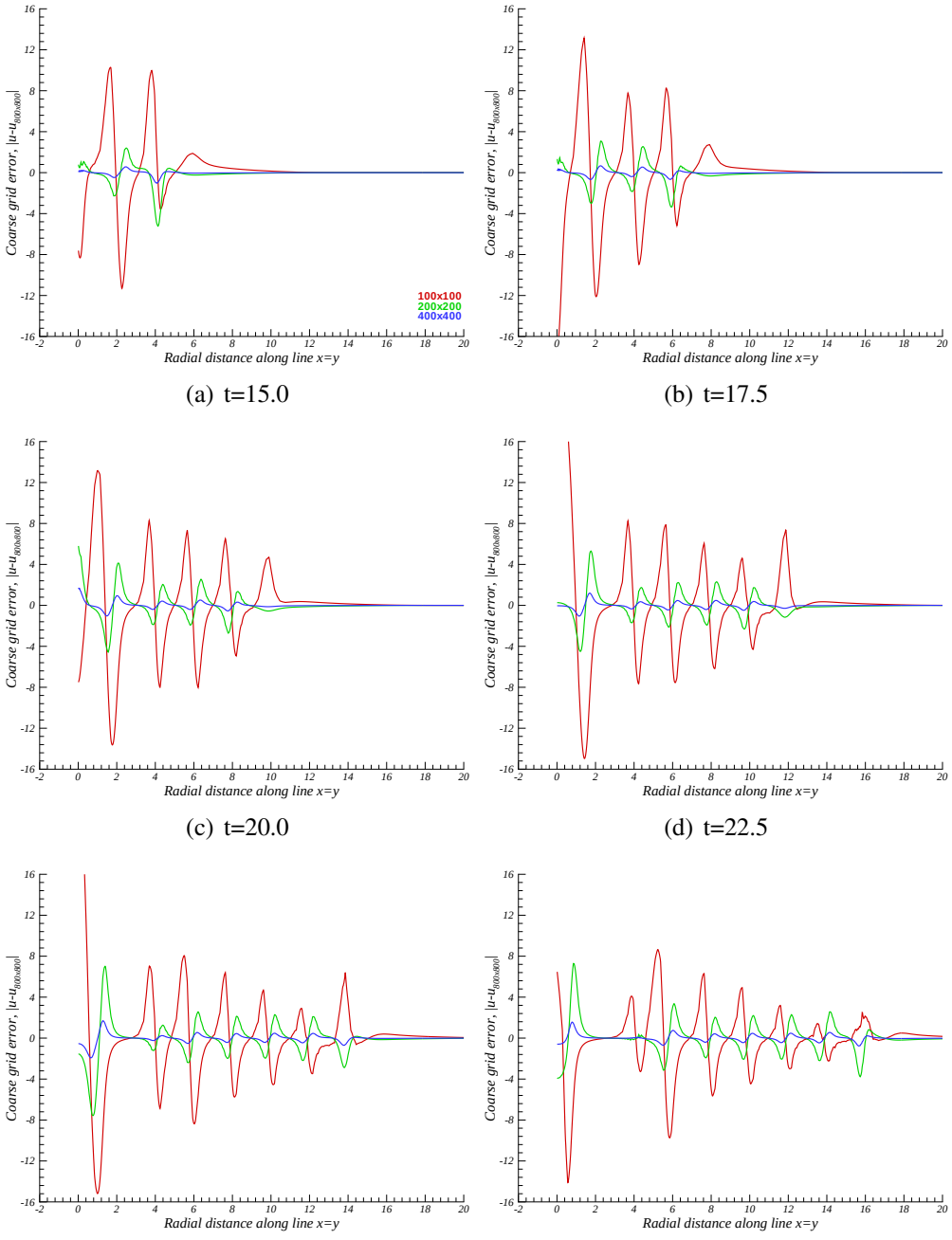
The figures reinforce the previous observation that the 100×100 results are completely erroneous. This is clear because the coarse grid error, $(u_{100\times100} - u_{800\times800})$ is of the same order of magnitude as the solution value itself, i.e. the coarse grid solution is not accurate to any significant digits.

The error for the 200×200 and 400×400 meshes is markedly decreased. Inspection of Figures 4.8 and 4.9 shows that the error for these two solutions follows the same trend. This is consistent with the previous observation that the solution is approaching mesh convergence for these resolutions.

4.4.1.4 Adaptive Mesh Refinement

It is clear from Figures 4.8–4.9 that the domain is largely quiescent for $t < 15$. For these early times the uniform Cartesian meshes considered previously are clearly "wasteful" since the fine mesh is only needed in the central, active subregion. To assess the viability of adaptive mesh refinement for this application the simulation was repeated with AMR beginning from a background 25×25 mesh. The maximum refinement level is restricted to four, which would correspond to a uniform 400×400 mesh. The adapted mesh is shown at two distinct times in Figure 4.10. The gradient indicator does an excellent job tracking the high curvature regions of the rings.

The number of degrees of freedom as a function of time step is shown in Figure 4.11. Recall that by limiting the element refinement to four levels a fine-grid spacing consistent with a 400×400 uniform Cartesian mesh results. Such a uniform mesh would result in a total of 482,403 degrees of freedom for the entire duration of the simulation. The tremendous savings afforded by the adaptive approach are clear from the figure. The largest problem size in the case of the adaptive mesh is more than a factor of two smaller than the equivalent uniform mesh. Further, for much of the simulated time the number of

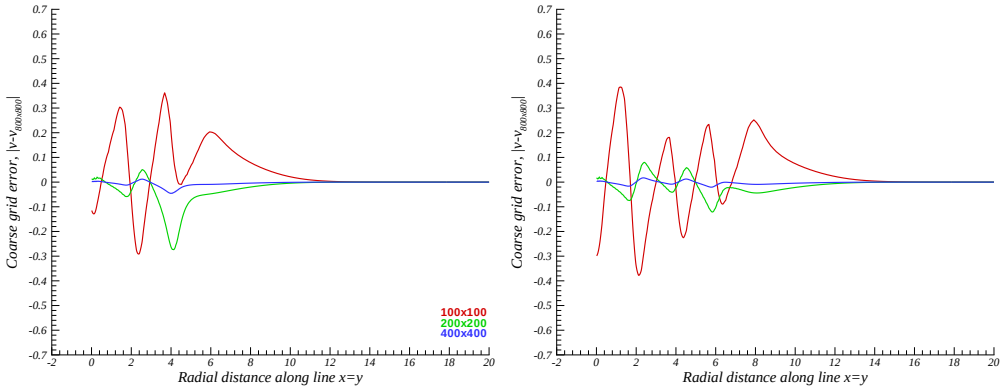


(a) t=15.0 (b) t=17.5

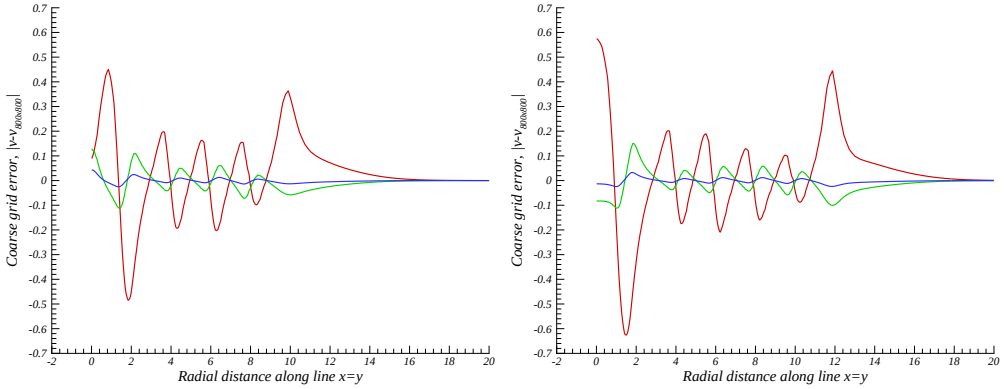
(c) t=20.0 (d) t=22.5

(e) t=25.0 (f) t=27.5

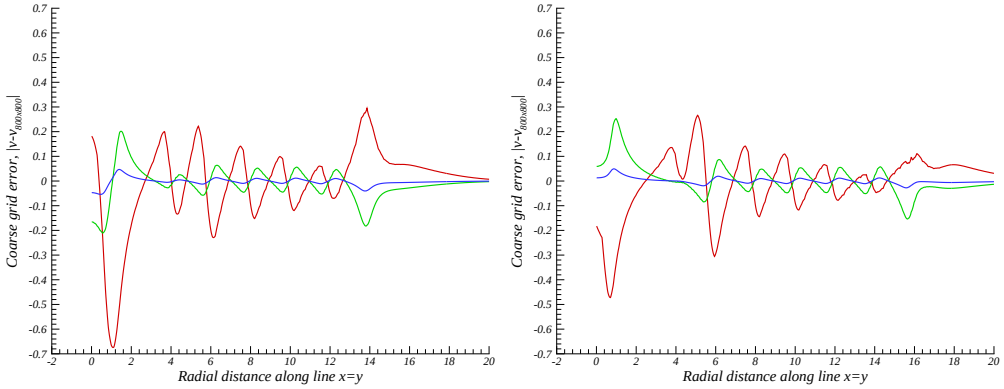
Figure 4.8: Continuous concentric advancing rings. Coarse grid bacteria concentration error history.



(a) t=15.0 (b) t=17.5



(c) t=20.0 (d) t=22.5



(e) t=25.0 (f) t=27.5

Figure 4.9: Continuous concentric advancing rings. Coarse grid chemoattractant concentration error history.

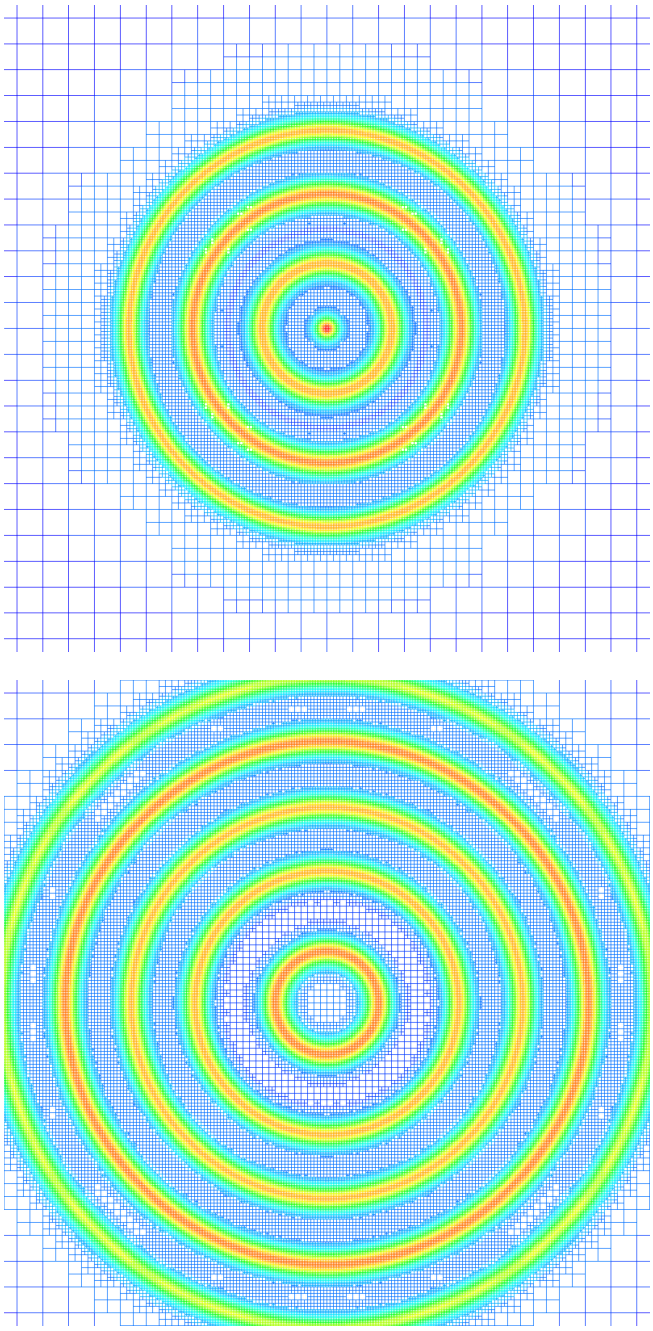


Figure 4.10: Continuous concentric advancing rings. Locally refined mesh for two instances in time. The mesh in the $[-10, 10]^2$ region of interest is colored by bacteria concentration.

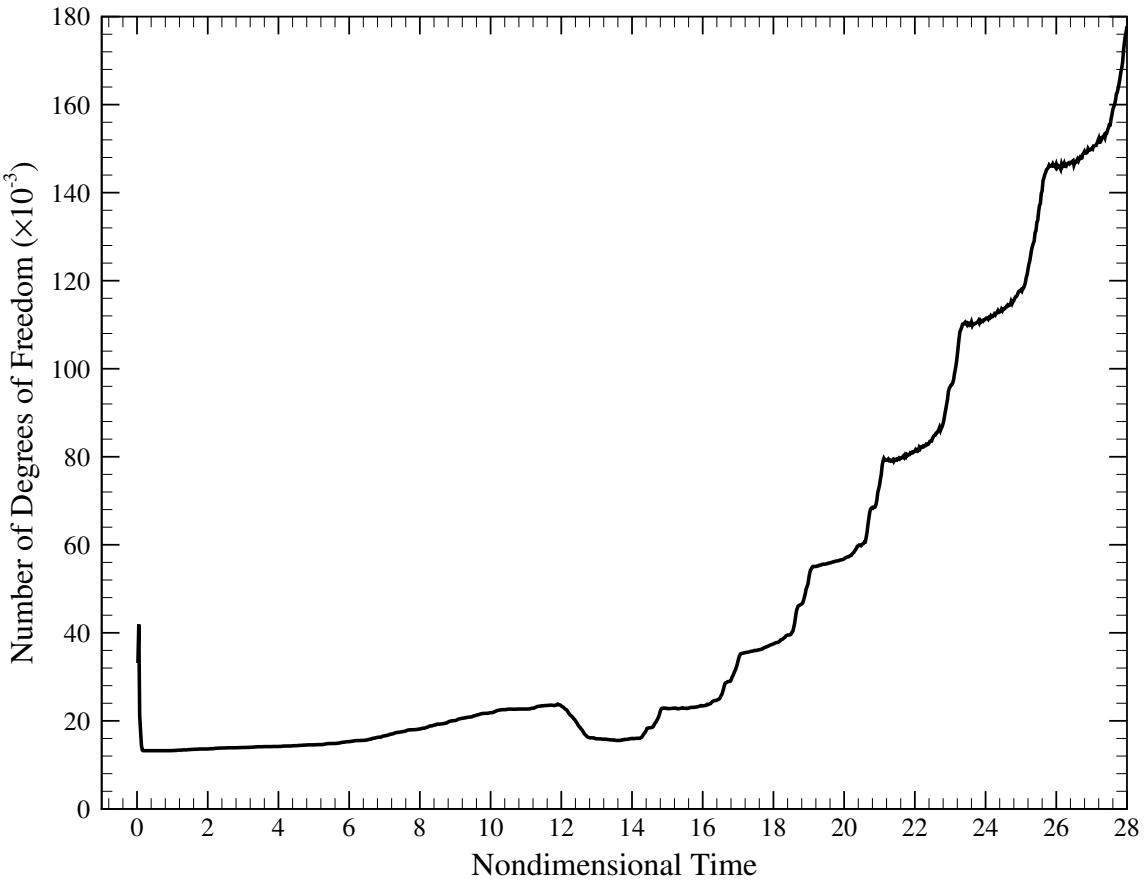


Figure 4.11: Continuous concentric advancing rings. Number of degrees of freedom as a function of time for the adaptive simulation.

degrees of freedom is less than 40,000 – a factor of *ten* fewer degrees of freedom than in the uniform mesh case.

4.4.2 Radial Spots Behind an Advancing Swarm Ring

4.4.2.1 Domain Specification and Initial Conditions

The initial conditions for this problem physically correspond to an initial inoculum of bacteria located at the center of a square domain which is defined as $\Omega = [-15, 15]^2$. As in the previous case, the domain is initially devoid of chemoattractant and has a uniform food source distribution. Also, symmetry is again assumed so that only a quarter-domain is simulated.

The initial nondimensional bacteria concentration for this case is $u_0 = 1$. As discussed in Section 4.4.1, it is again important that u_0 be specified in such a way that it can be implemented consistently across a range of mesh resolutions and element types. For this problem the same sequence of uniformly refined meshes with both bilinear and biquadratic elements to test mesh convergence. The coarsest mesh contains 100×100 elements, yielding a coarse mesh spacing of $h_c = 0.15$. For this family of meshes the initial conditions are taken as a tensor product of the following one-dimensional distributions:

$$u_0 = 1 \qquad x \leq h_c \tag{4.74}$$

$$= 2 - \frac{x}{h_c} \qquad h_c < x < 2hc \tag{4.75}$$

$$= 0 \qquad x \geq 2hc \tag{4.76}$$

$$v_0 = 0 \tag{4.77}$$

$$w_0 = \frac{8}{10} \tag{4.78}$$

Table 4.2: Nondimensional parameter values for radial spots deposited behind an advancing swarm ring (from Woodward et al. [21])

d_u	d_w	α	β	δ	ρ	μ	κ
1/4	1	30	10	7/2	1	50	0

4.4.2.2 System Parameters

The relevant parameters for this case were taken from Woodward et al. [21] and are listed in Table 4.2. The value $\alpha = 30$, which weights the chemotaxis term, is large enough for the system to exhibit appreciable chemotaxis-induced bacteria transport. Specifically for this choice of parameters, equations (4.17)–(4.19) become

$$\frac{\partial u}{\partial t} = \frac{1}{4}\Delta u - 30\ \boldsymbol{\nabla}\cdot\left[\frac{u}{(1+v)^2}\boldsymbol{\nabla}v\right] + \frac{7\ u}{2}\left(\frac{w^2}{1+w^2} - u\right) \tag{4.79}$$

$$\frac{\partial v}{\partial t} = \Delta v + \frac{10\ wu^2}{50+u^2} - uv \tag{4.80}$$

$$\frac{\partial w}{\partial t} = \Delta w \tag{4.81}$$

In particular, the parameter $\kappa = 0$ again decouples the stimulant concentration (w) from the bacterial (u) and chemoattractant (v) concentrations. The governing equations reduce to

$$w = \frac{8}{10} \tag{4.82}$$

$$\frac{\partial u}{\partial t} = \frac{1}{4}\Delta u - 30\ \boldsymbol{\nabla}\cdot\left[\frac{u}{(1+v)^2}\boldsymbol{\nabla}v\right] + u\left(\frac{56}{41} - u\right) \tag{4.83}$$

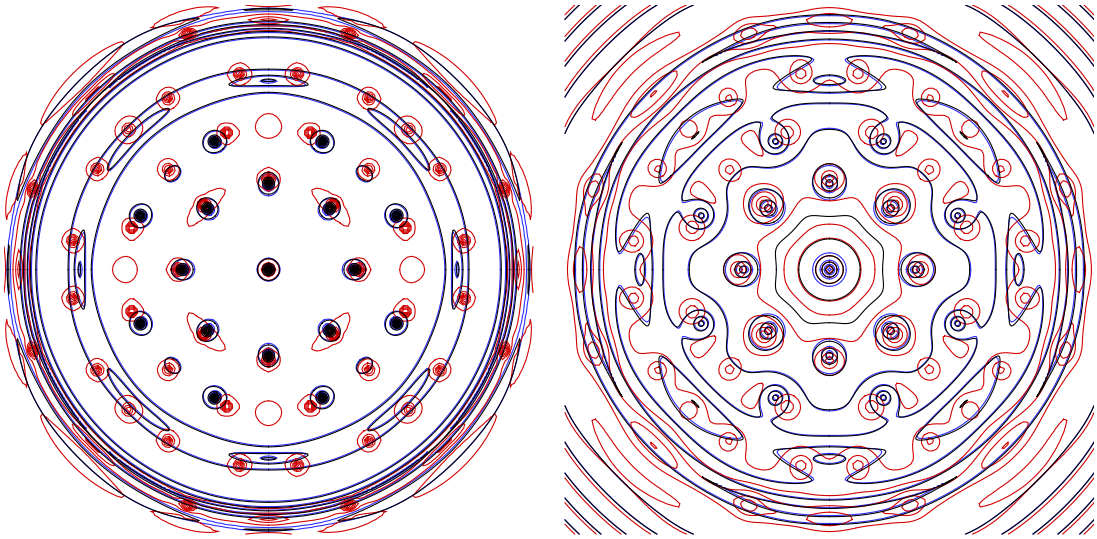
$$\frac{\partial v}{\partial t} = \Delta v + \frac{8\ u^2}{50+u^2} - uv \tag{4.84}$$

As mentioned previously, the decoupling of the governing equations when $\kappa = 0$ could be exploited numerically to reduce storage requirements.

4.4.2.3 Mesh and Time Convergence Studies

This section presents the results of mesh and time convergence studies. The goal of these studies was (i) to produce high-accuracy reference solutions and (ii) to determine the time accuracy required for the system. These solutions will be used for comparison to validate the adaptive algorithm. Additional simulations varied the approximation and quadrature orders to verify the accuracy of the scheme.

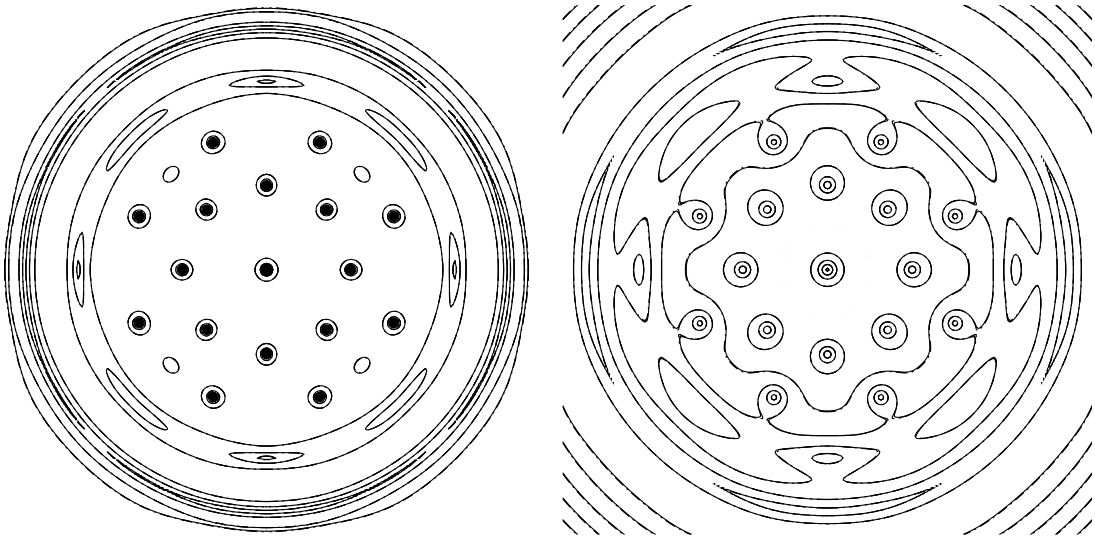
4.4.2.3.1 Mesh Convergence: Simulations were performed on a sequence of uniformly refined meshes to ensure mesh convergence of the solution. Figure 4.12 shows bacteria and chemoattractant contours for uniform meshes of 100×100, 200×200, and 400×400 elements. The negligible difference between the two finest meshes shows that mesh convergence is obtained and that a 200×200 element mesh is adequate for this problem.



(a) Bacteria (b) Chemoattractant

Figure 4.12: Overlaid concentrations at $t = 19$ illustrating mesh convergence for 100×100, 200×200, and 400×400 uniform meshes. The region of interest is a $[-10, 10]^2$ subdomain. Mesh convergence is obtained for the 200×200 and finer meshes.

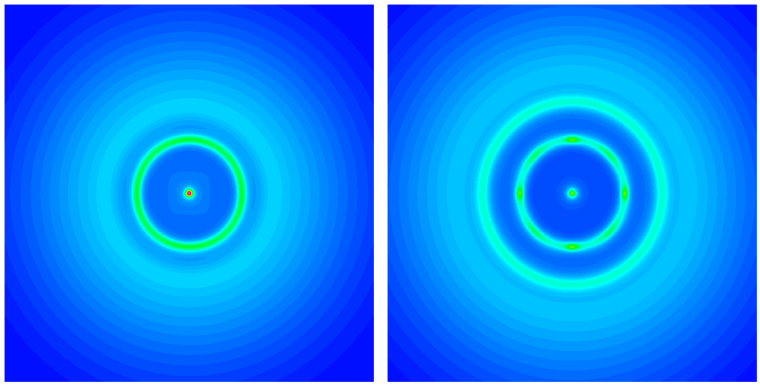
4.4.2.3.2 Time Convergence: Additional simulations were performed on a 200×200 element mesh to investigate the time convergence of the solution. Figure 4.13 shows bacteria and chemoattractant contours at a nondimensional time of $t = 19$. For these cases the time accuracy tolerance ε_t (Equation (4.51)) was varied. The figure shows that time convergence is obtained for values of $\varepsilon_t \leq 1 \times 10^{-2}$.



(a) Bacteria (b) Chemoattractant

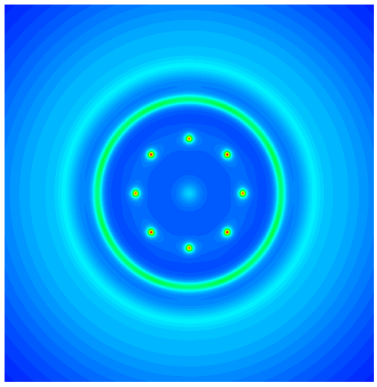
Figure 4.13: Overlaid concentrations at $t = 19$ on a 200×200 uniform mesh illustrating time convergence for ε_t $=2 \times 10^{-2}$, 1×10^{-2}, 5×10^{-3}, and 2.5×10^{-3}. The region of interest is a $[-10, 10]^2$ subdomain. Essentially all cases are time converged.

The maximum nondimensional bacteria and chemoattractant concentrations are plotted as a function of time for a series of hierarchically refined meshes in Figure 4.16.

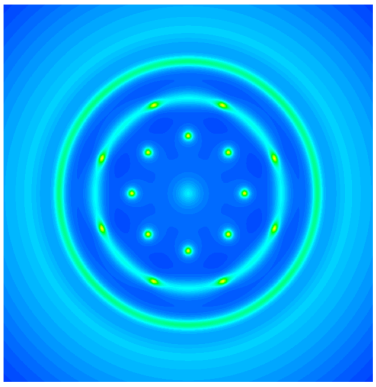


(a) t=12.50

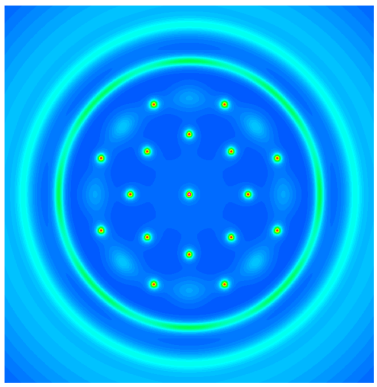
(b) t=13.75



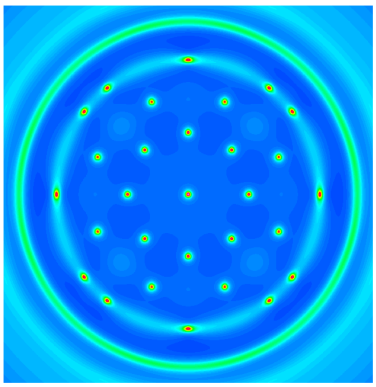
(c) t=15.00



(d) t=16.25

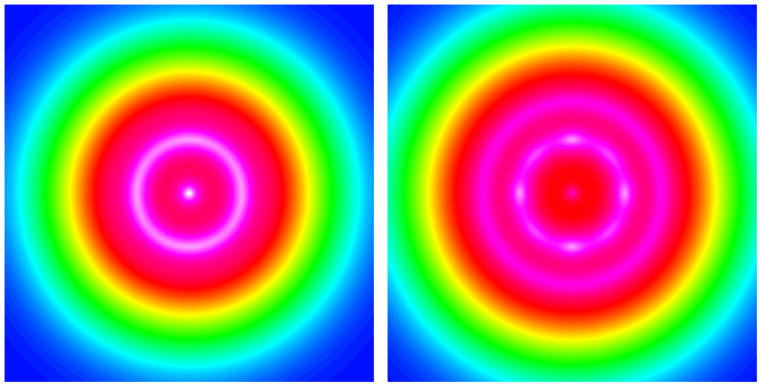


(e) t=17.50

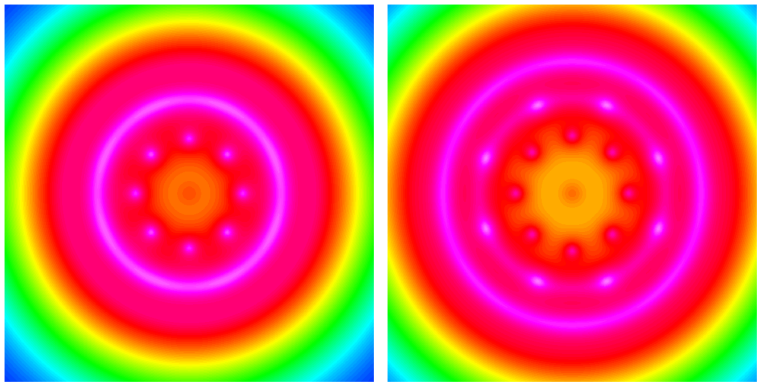


(f) t=18.75

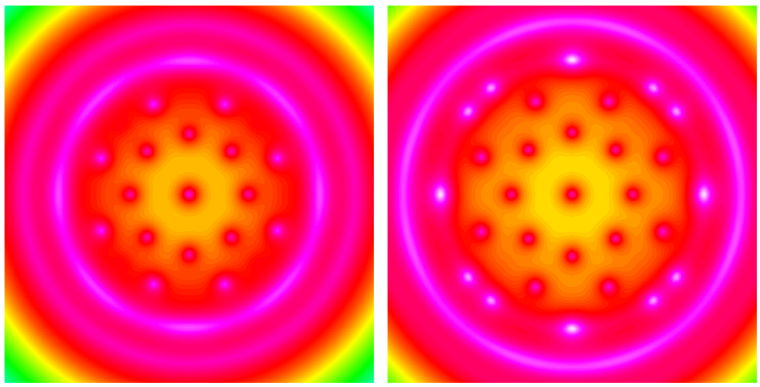
Figure 4.14: Radial spots. Bacteria concentration history. The region of interest is a $[-10, 10]^2$ subdomain.



(a) t=12.50 (b) t=13.75



(c) t=15.00 (d) t=16.25



(e) t=17.50 (f) t=18.75

Figure 4.15: Radial spots. Chemoattractant concentration history. The region of interest is a $[-10, 10]^2$ subdomain.

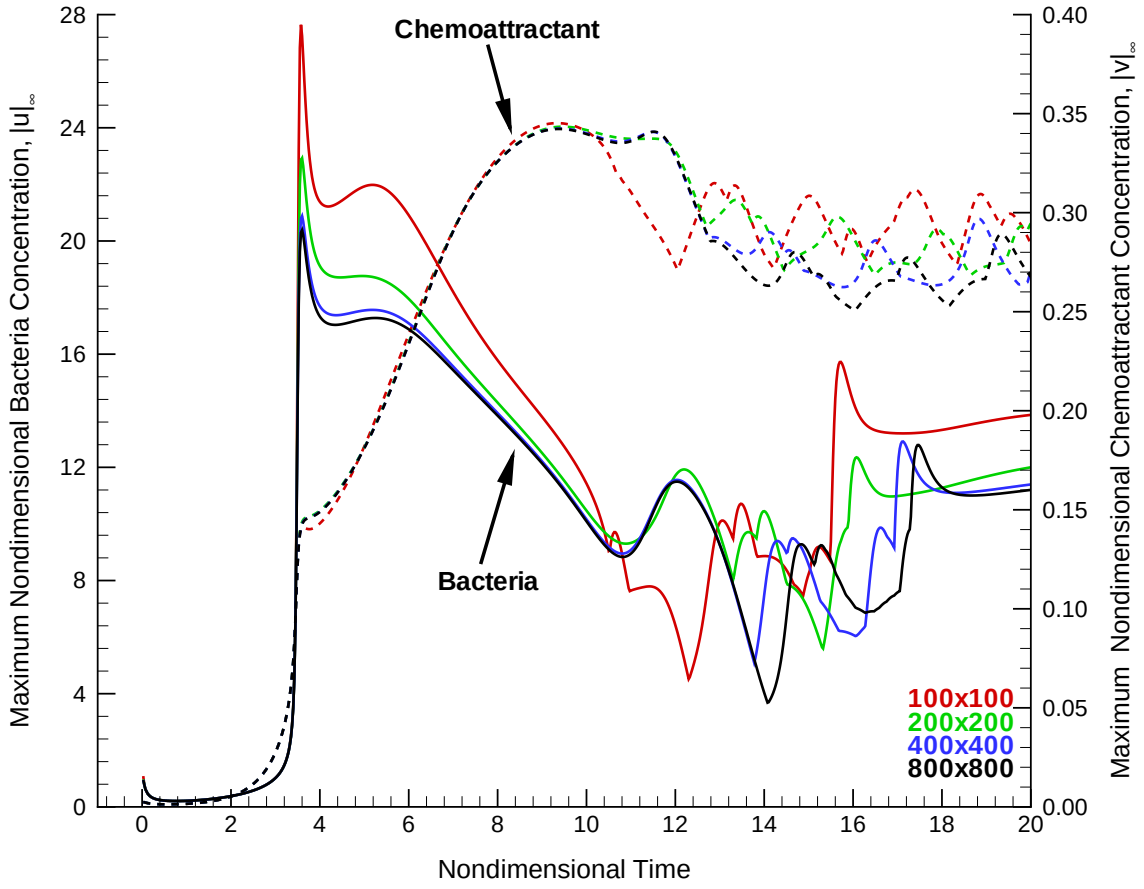


Figure 4.16: Radial spots. Maximum bacteria and chemoattractant history for a sequence of meshes.

4.4.2.4 Adaptive Mesh Refinement

As in the previous case, the simulation was repeated with AMR beginning from a background 25×25 mesh. The maximum refinement level is restricted to four, which would correspond to a uniform 400×400 mesh. The adapted mesh is shown at two distinct times in Figure 4.17.

The number of degrees of freedom as a function of time step is shown in Figure 4.18. As in the previous case, by limiting the maximum element refinement level to four a fine-grid spacing consistent with a 400×400 uniform Cartesian mesh results. Recall that such a uniform mesh would result in a total of 482,403 degrees of freedom for the entire duration of the simulation. Again, the largest problem size in the case of the adaptive mesh is more than a factor of two smaller than the equivalent uniform mesh. The degree of freedom history is distinctly different for this case than the previous one. Prior to $t = 12$ there is a steady, nearly monotone increase in the number of degrees of freedom. This is followed by a sharp decline as the "spots" begin to form in the domain. The number of degrees of freedom is then seen to increase in a sawtooth-like pattern as the number of degrees of freedom in the domain increases.

Returning to Figures 4.14–4.15 helps explain this behavior. Prior to $t = 12$ the domain is dominated by a large, smooth bacteria concentration which expands throughout the domain. As this concentration expands it begins to fill the domain, and there is no defining feature for the adaptive scheme to focus upon. Thus, the scheme refines the mesh nearly uniformly in the center of the domain where the bacteria are located. After this time, however, patterns of rings and spots begin to form. Since rings at later times are larger than the previous rings, more degrees of freedom are required to resolve them on this Cartesian mesh. The 'sawtooth" shape corresponds to the breakdown of a ring into a number of spots. During this process the features becomes increasingly more localized, hence a smaller number of degrees of freedom are required to resolve them to a specified resolution.

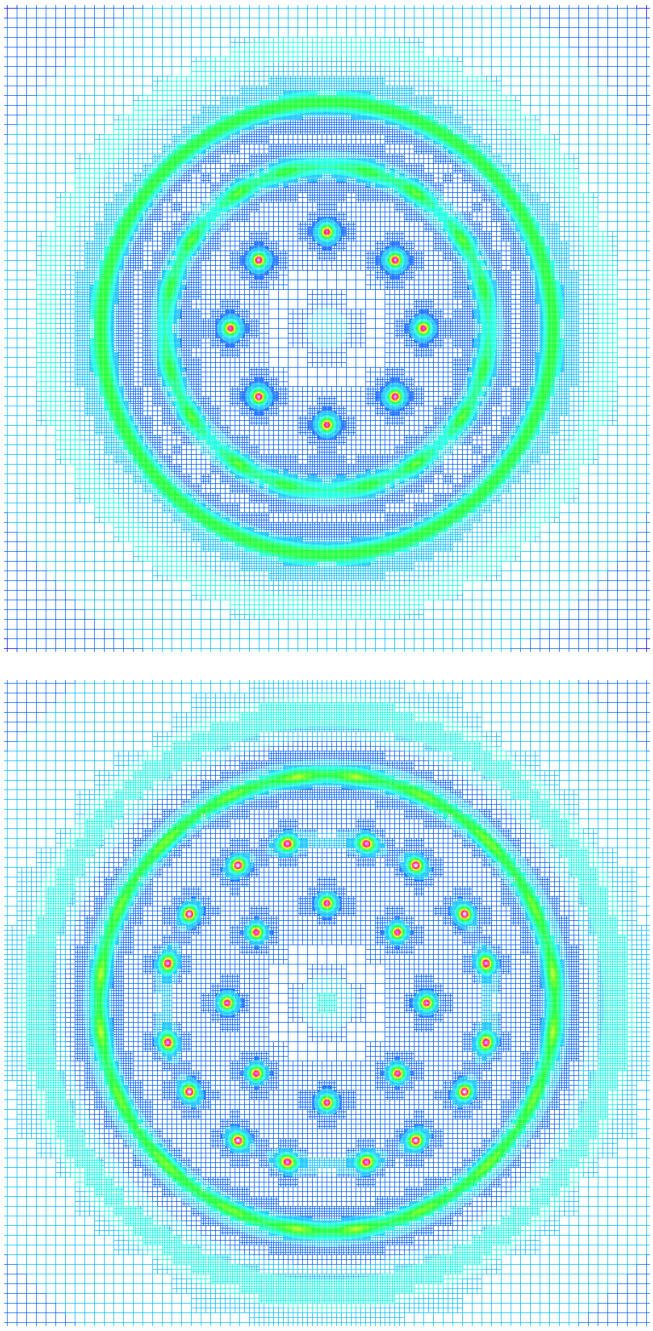


Figure 4.17: Continuous concentric advancing rings. Locally refined mesh for two instances in time. The mesh in the $[-10, 10]^2$ region of interest is colored by bacteria concentration.

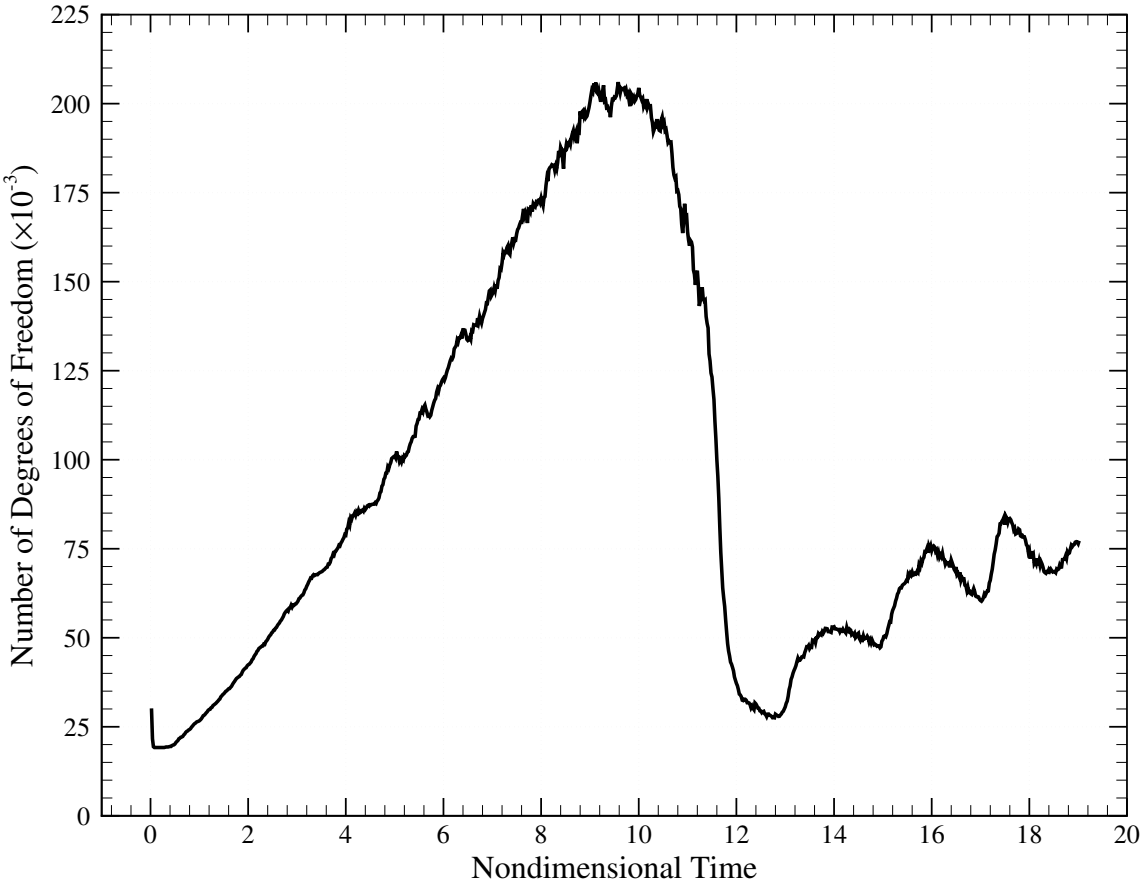


Figure 4.18: Radial spots. Number of degrees of freedom as a function of time for the adaptive simulation.

Chapter 5

Compressible Flows

5.1 Introduction

The goal of this chapter is to apply the parallel and adaptive simulation techniques presented previously to the particular problem of compressible flows. Compressible flows encompass a wide range of applications which of are of particular interest in the design and analysis of atmospheric flight and entry vehicles. An important parameter in characterizing compressible flows is the Mach number, M, which is the ratio of the fluid speed to the speed of sound in the medium. Compressible flowfields typically arise in one of four regimes [22, 23]:

1. Subsonic flows in which $M < 1$ everywhere
2. Transonic flows in which the majority of the flowfield is subsonic with the exception of localized regions
3. Supersonic flows in which $M > 1$ in the majority of the flowfield
4. Hypersonic flows in which it is generally assumed that $M > 5$

Compressible viscous flows with strong shock waves and viscous boundary layers are particularly well suited to simulation with adaptive techniques because of the disparate spatial scales involved in the flowfield. For example, a supersonic aircraft may be several tens of meters in length, but at cruise conditions the boundary layer enveloping the vehicle will be at most centimeters in thickness. Further, shock wave thickness is proportional to the mean free path in the gas, which is on the order of micrometers for air at sea level.

This chapter begins with the presentation of the compressible Navier–Stokes and Euler equations which are used to model this class of flows. A finite element formulation is then developed with the goal of applying the adaptive mesh solution techniques described previously. A fully implicit algorithm is used to preclude explicit stability restrictions which are dependent on mesh size. The time discretization and nonlinear solution techniques used in the computational algorithm are also described in detail.

The performance of the algorithm is then benchmarked and investigated with a series of two-dimensional viscous and inviscid flows. Of particular interest are the stability, consistency, and convergence properties of the current approach. More complicated three-dimensional flows are then considered for several high–lift reentry vehicle configurations, including the Space Shuttle Orbiter. Since the primary goal of this chapter is to assess the suitability of adaptive techniques for multiscale spatial behavior of compressible viscous flowfields, the problems considered here are restricted to laminar, perfect gas flows. The observations made with respect to adaptive algorithms for treating these spatial scale issues will generalize to flows with other constitutive gas models and compressible flow classes.

Some particularly challenging problems involve complex interactions of shock waves with each other and shock waves with boundary layers. The viscous fluid dynamic boundary layer may also be accompanied by thermal boundary layers in coupled heat transfer problems. These then typify the complex class of multiphysics/multiscale problems for which AMR strategies are particularly relevant but still exhibit some interesting challenges and open questions that need to be addressed. For instance, this work studies shock/boundary layer and shock/shock interaction problems which occur in hypersonic aerothermodynamic applications and compute AMR solutions for surface heat transfer and pressure distributions.

5.2 Mathematical Model

The compressible Navier–Stokes equations describe the conservation of mass, momentum, and energy for this class of flows. This section reviews the Navier–Stokes system of equations, relevant state equations and transport property models for air, and provides the nondimensionalization scheme which is used in the present work.

5.2.1 Governing Equations

The conservation of mass, momentum, and energy for a compressible fluid may be written as

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u}) = 0 \tag{5.1}$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\boldsymbol{\nabla} P + \boldsymbol{\nabla} \cdot \boldsymbol{\tau} \tag{5.2}$$

$$\frac{\partial \rho E}{\partial t} + \boldsymbol{\nabla} \cdot (\rho E \boldsymbol{u}) = -\boldsymbol{\nabla} \cdot \boldsymbol{q} - P \boldsymbol{\nabla} \cdot \boldsymbol{u} + \boldsymbol{\nabla} \cdot (\boldsymbol{\tau} \boldsymbol{u}) \tag{5.3}$$

where ρ is the density, $\boldsymbol{u}$ is the velocity, E is the total energy per unit mass, and P is the pressure. The total energy per unit mass, E, in Equation (5.3) may be decomposed into internal and kinetic contributions

$$E = e + \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2} \tag{5.4}$$

and the total enthalpy per unit mass, H, is given by

$$H = E + \frac{P}{\rho} \tag{5.5}$$

The viscous stress tensor $\boldsymbol{\tau}$ and the heat flux vector $\boldsymbol{q}$ are defined as

$$\boldsymbol{\tau} = \mu \left(\boldsymbol{\nabla} \boldsymbol{u} + \boldsymbol{\nabla}^T \boldsymbol{u} \right) + \lambda \left(\boldsymbol{\nabla} \cdot \boldsymbol{u} \right) \boldsymbol{I} \tag{5.6}$$

$$\boldsymbol{q} = -k \boldsymbol{\nabla} T \tag{5.7}$$

where μ is the dynamic viscosity, λ is the second coefficient of viscosity, k is the thermal conductivity, and T is the fluid temperature. The two coefficients of viscosity are related to

the bulk viscosity κ by

$$\kappa = \frac{2}{3}\mu + \lambda \tag{5.8}$$

In general, the bulk viscosity is negligible except in detailed studies of shock wave structure or for investigations of the adsorption and attenuation of acoustic waves [24]. Under this assumption, $\kappa = 0$ in Equation (5.8) and λ is defined as

$$\lambda = -\frac{2}{3}\mu \tag{5.9}$$

Equation (5.6) with (5.9) is known as Stokes' hypothesis for a Newtonian fluid.

5.2.2 Equations of State

In three dimensions Equations (5.1)–(5.3) provide a system of five coupled partial differential equations in the seven unknowns $\rho, \boldsymbol{u}, e, P, T$, provided that the transport properties μ and k may be related to the unknown thermodynamic properties. Clearly, two additional equations are required to close the system. These additional equations are equations of state that relate the thermodynamic variables ρ, e, P, T. Assuming that the fluid is in chemical equilibrium, the state principle of thermodynamics dictates that the thermodynamic state of a system is fixed by any two independent thermodynamic variables. Thus, by choosing ρ and e to be the independent variables, state equations for $P = P(\rho, e)$ and $T = T(\rho, e)$ may be obtained.

For many problems in gas dynamics intermolecular forces within the gas are negligible. Such gases are governed by the perfect gas equation of state

$$P = \rho R T \tag{5.10}$$

where R is the gas constant and is equal to 286.9 $m^2/(s^2\ K)$ for air at standard conditions.

Perfect gases at relatively low temperatures may also considered calorically perfect. In a calorically perfect gas the specific heats at constant pressure and volume, c_p and c_v respectively, are constant. The ratio of specific heats $\gamma = c_p/c_v$ is also constant, and for

air at standard temperatures $\gamma = 1.4$. The specific heats c_p, c_v are then related to the gas constant R by

$$c_p = \frac{\gamma R}{\gamma - 1} \tag{5.11}$$

$$c_v = \frac{R}{\gamma - 1} \tag{5.12}$$

For calorically perfect gases the internal energy and enthalpy are directly proportional to the temperature:

$$e = c_v T \tag{5.13}$$

$$h = c_p T \tag{5.14}$$

Combining Equations (5.10)–(5.14) yields the desired equations of state

$$P = (\gamma - 1)\,\rho e \tag{5.15}$$

$$T = \frac{(\gamma - 1)\,e}{R} \tag{5.16}$$

For air at moderate temperatures the calorically perfect gas assumption breaks down. At these temperatures the oxygen and nitrogen molecules begin to exhibit vibrational excitation, and the specific heats c_p, c_v are no longer constant but instead become functions of temperature. Such gases are said, by definition, to be thermally perfect. Air becomes thermally perfect at temperatures exceeding approximately 800 K. For thermally perfect gases the perfect gas equation of state (Equation (5.10)) still holds, but the temperature is no longer directly proportional to the internal energy as specified in Equation (5.16). For such gases the state equations $P = P(\rho, e)$ and $T = T(\rho, e)$ can be obtained from tables, charts, or curve fits [25].

As the gas temperature is increased further the vibrationally excited molecules begin to dissociate. Air, which is primarily composed of N_2 and O_2 molecules, starts to dissociate around 2000 K. At this temperature the molecular oxygen starts breaking down into atomic oxygen ($O_2 \rightarrow 2O$), and essentially all the molecular oxygen is dissociated

by 4000 K. At this temperature the stronger bonds in the molecular nitrogen begin to break ($N_2 \rightarrow 2N$), resulting in atomic nitrogen. Additionally, nitric oxide NO can then be formed. Essentially all of the molecular nitrogen is dissociated by 9000 K, and further increases in temperature produce ionization.

The application studies presented in Section 5.6 are limited to flows for which the calorically perfect gas assumption is valid, hence chemically reacting flows will not be considered further. For atmospheric flight the calorically perfect gas assumption is valid for Mach numbers on the order of 5 or less. However, in the case of experimental wind tunnel data, high Mach number perfect gas flows may be achieved by lowering the freestream static temperature (and hence speed of sound) rather than increasing freestream velocity. In such a case the flowfield may remain calorically perfect at freestream Mach numbers exceeding 10. Section 5.6 will present comparisons with wind tunnel data at Mach numbers as high as 12.5.

5.2.3 Transport Properties

The remaining coefficients of viscosity and thermal conductivity may be related to the thermodynamic variables using kinetic theory. For air over a wide range of temperatures, $\mu = \mu(T)$ and is given by Sutherland's law [26]

$$\mu_{\text{air}} = 1.458 \times 10^{-6} \frac{T^{\frac{3}{2}}}{T + 110.4} \ \text{Pa} \cdot \text{s} \tag{5.17}$$

where T is in Kelvin. With the viscosity given by Equation (5.17) for a given temperature, it is convenient to determine the thermal conductivity k assuming a constant Prandtl number. The Prandtl number, which defines the ratio of the fluid's viscous to thermal diffusivity, is defined as

$$Pr = \frac{\mu c_p}{k} \tag{5.18}$$

and $Pr = 0.72$ for air at standard conditions.

For chemically reacting flows more complicated models are required for the transport properties. One popular model, presented by Gupta et al. [27] uses binary collision–

integral based mixing rules, which provide a good approximation of reacting flow transport properties over a range of flow conditions. The present studies will be restricted to flows in which (5.17) and the assumption of constant Prandtl number hold.

5.2.4 Nondimensionalization

For the iterative implicit algorithms employed in the present work it is desirable that the unknown solution values are of the same order of magnitude for all solution components. Disparate orders of magnitude in individual components can cause small changes in one variable to mask relatively large changes in another when evaluating vector norms. This is an important consideration, for example, when using vector norms to monitor solution convergence.

The governing equations (5.1)–(5.3) can be nondimensionalized in a number of ways. The present work employs the Reynolds number as the basis for the nondimensionalization. The Reynolds number describes the ratio of convective to diffusive forces within a flow is given by

$$Re_L = \frac{\rho_\infty U_\infty L}{\mu_\infty} \tag{5.19}$$

where L is a reference length and $()_\infty$ denotes freestream values. The independent variables are then nondimensionalized as follows:

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x}}{L} \tag{5.20}$$

$$\hat{\boldsymbol{u}} = \frac{\boldsymbol{u}}{U_\infty} \tag{5.21}$$

$$\hat{t} = \frac{t}{L/U_\infty} \tag{5.22}$$

$$\hat{\rho} = \frac{\rho}{\rho_\infty} \tag{5.23}$$

$$\hat{p} = \frac{p}{\rho_\infty U_\infty^2} \tag{5.24}$$

$$\hat{T} = \frac{T}{T_\infty} \tag{5.25}$$

$$\hat{e} = \frac{e}{U_\infty^2} \tag{5.26}$$

$$\hat{\mu} = \frac{\mu}{Re_L\ \mu_\infty} \tag{5.27}$$

Substituting (5.20)–(5.27) into (5.1)–(5.3) yields an unchanged set of equations, which is convenient for developing a numerical scheme which may be employed for both dimensional and nondimensional simulations [28].

5.2.5 System of Equations

Equations (5.1)–(5.3) may be written in system form as

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_i}{\partial x_i} = \frac{\partial \boldsymbol{G}_i}{\partial x_i} \tag{5.28}$$

where the vector $\boldsymbol{U}$ consists of the so-called conservation variables, $\boldsymbol{F}_i$ and $\boldsymbol{G}_i$ are the inviscid and viscous fluxes in the i^{th} direction, respectively. The conservation variables $\boldsymbol{U}$ in three dimensions are defined as

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix} \tag{5.29}$$

correspond to the fluid density, Cartesian components of momentum per unit volume, and total energy per unit volume, respectively. The inviscid flux $\boldsymbol{F}_i$ in (5.28) is given by

$$\boldsymbol{F}_i = \begin{bmatrix} \rho u_i \\ \rho u_i u_1 + \delta_{i1} P \\ \rho u_i u_2 + \delta_{i2} P \\ \rho u_i u_3 + \delta_{i3} P \\ \rho u_i H \end{bmatrix} \tag{5.30}$$

where δ_{ij} is the Kronecker delta satisfying $\delta_{ij} = 0$ when $i \neq j$ and is of unit value otherwise. The viscous flux is

$$\boldsymbol{G}_i = \begin{bmatrix} 0 \\ \tau_{i1} \\ \tau_{i2} \\ \tau_{i3} \\ -q_i + \tau_{ik} u_k \end{bmatrix} \tag{5.31}$$

The second term on the left-hand-side of (5.28) is the divergence of the inviscid flux vector, $\frac{\partial \boldsymbol{F}_i}{\partial x_i}$, and may be written in terms of the unknowns $\boldsymbol{U}$ as

$$\frac{\partial \boldsymbol{F}_i}{\partial x_i} = \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{U}} \frac{\partial \boldsymbol{U}}{\partial x_i} = \boldsymbol{A}_i \frac{\partial \boldsymbol{U}}{\partial x_i} \tag{5.32}$$

where $\boldsymbol{A}_i = \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{U}}$ is the inviscid flux Jacobian. Similarly, the viscous flux vector $\boldsymbol{G}_i$ may be written as

$$\frac{\partial \boldsymbol{G}_i}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\boldsymbol{K}_{ij} \frac{\partial \boldsymbol{U}}{\partial x_j} \right) \tag{5.33}$$

where $\boldsymbol{K}_{ij}$ is the viscous flux Jacobian. The Jacobian matrices $\boldsymbol{A}_i$ and $\boldsymbol{K}_{ij}$ are both functions of the independent variables $\boldsymbol{U}$ and are defined in Appendix A.1. Using (5.32) and (5.33) in (5.28) yields the second-order system

$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}_i \frac{\partial \boldsymbol{U}}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\boldsymbol{K}_{ij}\frac{\partial \boldsymbol{U}}{\partial x_j}\right) \tag{5.34}$$

which will be the basis for developing a weak formulation in Section 5.3.

5.3 Weak Formulation

5.3.1 Galerkin Weak Statement

The corresponding weak form of the governing system of Equations (5.34) is obtained by first multiplying by an appropriate set of test functions $\boldsymbol{W}$ and integrating over the domain Ω to obtain

$$\int_\Omega \boldsymbol{W} \cdot \left[\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}_i \frac{\partial \boldsymbol{U}}{\partial x_i} - \frac{\partial}{\partial x_i}\left(\boldsymbol{K}_{ij}\frac{\partial \boldsymbol{U}}{\partial x_j}\right)\right] \, d\Omega = 0 \tag{5.35}$$

or, after integrating the last term in (5.35) by parts,

$$\int_\Omega \left[\boldsymbol{W} \cdot \left(\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}_i \frac{\partial \boldsymbol{U}}{\partial x_i}\right) + \frac{\partial \boldsymbol{W}}{\partial x_i} \cdot \left(\boldsymbol{K}_{ij}\frac{\partial \boldsymbol{U}}{\partial x_j}\right)\right] \, d\Omega - \int_\Gamma \boldsymbol{W} \cdot \boldsymbol{g} \, d\Gamma = 0 \tag{5.36}$$

where $\boldsymbol{g} = \boldsymbol{G} \cdot \hat{\boldsymbol{n}}$ is the normal component of the viscous flux on the boundary Γ with unit normal $\hat{\boldsymbol{n}}$. In the inviscid limit we recover the first-order Euler equations (discussed later in detail) and the system of partial differential equations is hyperbolic.

5.3.2 Stabilized Formulation

A standard Galerkin finite element formulation as presented in (5.36) (or similar finite difference or finite volume strategies) is unstable in the sense that it may produce nonphysical oscillations in regions of steep solution gradients. Even when viscous effects are included as in (5.36) standard Galerkin calculations will produce non-physical oscillations. This phenomenon results because the standard Galerkin formulation (or equivalently

central differencing on a structured grid) produces an odd-even decoupling between adjacent nodes in the solution, hence the discretized solution admits oscillatory behavior.

For some classes of flow and transport this instability can be related to inadequate spatial resolution in the grid. In these cases the Galerkin discretization on a sufficiently refined mesh will produce stable results. This is typically the case for low-speed incompressible flows for which there is an approximate balance between the convective and diffusive length scales. This balance is described by the cell Reynolds number, which is defined as

$$Re_c \equiv \frac{\rho\, U\, h_{ref}}{\mu} \tag{5.37}$$

where h_{ref} is the cell reference length and the other properties are evaluated locally. When the local flow properties and mesh spacing is such that $\mathrm{Re}_c < 2$ the standard Galerkin formulation will yield non-oscillatory results. Unfortunately, such a balance is rarely achieved for compressible flows in aerospace applications. Indeed, the Euler equations are devoid of any diffusion, so a standard Galerkin discretization such as in Equation (5.36) will always exhibit stability issues, regardless of mesh resolution.

Several techniques have been proposed to address the stability issue of the Galerkin formulation. The familiar Lax–Wendroff finite difference scheme produces the Taylor–Galerkin scheme in the context of finite elements. The Taylor–Galerkin scheme employs a second-order Taylor series in time and an interchange of spatial and temporal differentiation in the discretization of (5.28). This yields a second–order term in the discrete form that can be interpreted as a stabilizing diffusion. Recently the Taylor–Galerkin scheme has been applied to hypersonic flowfields in chemical and thermal nonequilibrium [29], illustrating its applicability to the class of problems considered in the present work.

A different approach is pursued by Carey et al. in the Least–Squares finite element method. In the Least–Squares approach the test function $\boldsymbol{W}$ in (5.35) is replaced by the variation of the residual of the governing equations [30, 31]. Conceptually this is equivalent to minimizing the residual in a least–squares sense. A detailed analysis of this formulation reveals a stabilizing mechanism similar to the Taylor–Galerkin scheme. This least–squares

idea can be combined with the Galerkin statement to yield the so-called Galerkin/least–squares scheme [32].

The stabilization introduced via numerical dissipation in upwind differencing can be achieved in the finite element setting when an upwind bias is added to the test function $\boldsymbol{W}$. This idea, and the need to reduce cross-wind dissipation in two or three dimensions, led to the development of the directed streamline–upwind Petrov/Galerkin (SUPG) formulation as another stabilizing mechanism for convection dominated flows [33]. For the system of equations (5.34) a suitably upstream-biased test function can be defined by augmenting the standard Galerkin test function $\boldsymbol{W}$ with the convective operator acting on the test function:

$$\begin{aligned} \hat{\boldsymbol{W}} &= \boldsymbol{W} + \boldsymbol{\tau}_{SUPG}\, \mathcal{L}_c\left(\boldsymbol{W}\right) \\ \hat{\boldsymbol{W}} &= \boldsymbol{W} + \boldsymbol{\tau}_{SUPG}\, \boldsymbol{A}_i \frac{\partial \boldsymbol{W}}{\partial x_i} \end{aligned} \tag{5.38}$$

The stabilization matrix $\boldsymbol{\tau}_{SUPG}$ plays an important role in the SUPG formulation in that it seeks to introduce the minimal amount of diffusion necessary to stabilize the scheme. In this work $\boldsymbol{\tau}_{SUPG}$ is adapted from previous work by Shakib in the context of entropy variables and later used by Aliabadi with the conservation variables [34, 35]. Specifically, in three dimensions

$$\boldsymbol{\tau}_{SUPG} = \text{diag}\left(\tau_c, \tau_m, \tau_m, \tau_m, \tau_e\right) \tag{5.39}$$

where τ_c, τ_m, and τ_e are scalar stabilization parameters for the continuity, momentum, and energy equations, respectively. These values are defined as

$$\begin{aligned} \tau_c &= \left[\left(\frac{2}{\Delta t}\right)^2 + \left(\frac{2\left(\|\boldsymbol{u}\| + c\right)}{h_{\boldsymbol{u}}}\right)^2\right]^{-1/2} \\ \tau_m &= \left[\left(\frac{2}{\Delta t}\right)^2 + \left(\frac{2\left(\|\boldsymbol{u}\| + c\right)}{h_{\boldsymbol{u}}}\right)^2 + \left(\frac{4\mu}{\rho h_{\boldsymbol{u}}^2}\right)^2\right]^{-1/2} \\ \tau_e &= \left[\left(\frac{2}{\Delta t}\right)^2 + \left(\frac{2\left(\|\boldsymbol{u}\| + c\right)}{h_{\boldsymbol{u}}}\right)^2 + \left(\frac{4k}{\rho c_p h_{\boldsymbol{u}}^2}\right)^2\right]^{-1/2} \end{aligned}$$

and are designed to transition smoothly between convective, diffusive, and transient-dominated flow regimes. The flow aligned element length scale, $h_{\boldsymbol{u}}$, is defined as

$$h_{\boldsymbol{u}} = 2\left(\sum_{k=1}^{\mathrm{NN}} |\hat{\boldsymbol{u}} \cdot \boldsymbol{\nabla}\phi_k|\right)^{-1}$$

where NN is the number of nodes in the element, $\{\boldsymbol{\nabla}\phi\}$ are the element shape function gradients, and $\hat{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$ is a the flow-aligned unit vector.

The SUPG weak statement then follows by multiplying (5.34) by (5.38) and integrating by parts as before

$$\begin{aligned}
&\int_\Omega \left[\boldsymbol{W}\cdot\left(\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}_i\frac{\partial \boldsymbol{U}}{\partial x_i}\right) + \frac{\partial \boldsymbol{W}}{\partial x_i}\cdot\left(\boldsymbol{K}_{ij}\frac{\partial \boldsymbol{U}}{\partial x_j}\right)\right]\, d\Omega \\
&+\sum_{e=1}^{n_{el}}\int_{\Omega_e} \boldsymbol{\tau}_{SUPG}\frac{\partial \boldsymbol{W}}{\partial x_k}\cdot\boldsymbol{A}_k\left[\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}_i\frac{\partial \boldsymbol{U}}{\partial x_i} - \frac{\partial}{\partial x_i}\left(\boldsymbol{K}_{ij}\frac{\partial \boldsymbol{U}}{\partial x_j}\right)\right]\, d\Omega \\
&- \int_\Gamma \boldsymbol{W}\cdot\boldsymbol{g}\, d\Gamma = 0
\end{aligned} \tag{5.40}$$

It is important to note that all of the schemes discussed previously address the convection–induced instability of the Galerkin finite element method. For supersonic problems involving strong shock waves another form of stabilization is required. More specifically, a local regularization scheme using a shock–capturing operator is required to eliminate nonphysical over and under–shoots induced by strong gradients. For this class of problems we augment (5.40) with an additional form of artificial diffusion to obtain:

$$\begin{aligned}
&\int_\Omega \left[\boldsymbol{W}\cdot\left(\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}_i\frac{\partial \boldsymbol{U}}{\partial x_i}\right) + \frac{\partial \boldsymbol{W}}{\partial x_i}\cdot\left(\boldsymbol{K}_{ij}\frac{\partial \boldsymbol{U}}{\partial x_j}\right)\right]\, d\Omega \\
&+\sum_{e=1}^{n_{el}}\int_{\Omega_e} \boldsymbol{\tau}_{SUPG}\frac{\partial \boldsymbol{W}}{\partial x_k}\cdot\boldsymbol{A}_k\left[\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}_i\frac{\partial \boldsymbol{U}}{\partial x_i} - \frac{\partial}{\partial x_i}\left(\boldsymbol{K}_{ij}\frac{\partial \boldsymbol{U}}{\partial x_j}\right)\right]\, d\Omega \\
&+\sum_{e=1}^{n_{el}}\int_{\Omega_e} \delta\left(\frac{\partial \boldsymbol{W}}{\partial x_i}\cdot\frac{\partial \boldsymbol{U}}{\partial x_i}\right)\, d\Omega - \int_\Gamma \boldsymbol{W}\cdot\boldsymbol{g}\, d\Gamma = 0
\end{aligned} \tag{5.41}$$

The shock capturing parameter is local and essentially regularizes the problem by selectively introducing isotropic artificial diffusion. This added local dissipation captures shocks

approximately across a few mesh cells. Note that by introducing the shock–capturing operator, δ, consistency with (5.34) is lost. Consequently, for flows with strong shocks the solution may exhibit some loss of accuracy in regions of large δ.

The shock capturing operator, δ, was adapted for a system of conservation variables by LeBeau and Tezduyar [34, 35, 36] from the original definition employed by Hughes et al. for the case of entropy variables [37, 38]. A modified form is employed in the present work and is defined as

$$\delta = \left[\frac{\left\| \frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}_i \frac{\partial \boldsymbol{U}}{\partial x_i} - \frac{\partial}{\partial x_i}\left(\boldsymbol{K}_{ij} \frac{\partial \boldsymbol{U}}{\partial x_j} \right) \right\|_{\boldsymbol{A}_0^{-1}}}{\|\boldsymbol{\nabla}\xi \cdot \boldsymbol{\nabla}\boldsymbol{U}\|_{\boldsymbol{A}_0^{-1}} + \|\boldsymbol{\nabla}\eta \cdot \boldsymbol{\nabla}\boldsymbol{U}\|_{\boldsymbol{A}_0^{-1}} + \|\boldsymbol{\nabla}\zeta \cdot \boldsymbol{\nabla}\boldsymbol{U}\|_{\boldsymbol{A}_0^{-1}}} \right]^{1/2} \tag{5.42}$$

where (ξ, η, ζ) are the canonical reference element coordinates and $\boldsymbol{A}_0^{-1}$ is the mapping from conservation to entropy variables (defined in Appendix A.2). The vector norm $\|\boldsymbol{v}\|_{\boldsymbol{A}_0^{-1}}$ is defined as $\boldsymbol{v}^T \left(\boldsymbol{A}_0^{-1} \boldsymbol{v} \right)$. The time derivative term was absent in the original formulations and has been added here for use in time-accurate simulations. Additionally, the diffusive term in the numerator is included so that consistency with (5.34) is maintained. That is, this form of the shock capturing parameter will vanish when the discrete solution satisfies (5.34).

The physical-domain to reference-domain element transformation terms $(\boldsymbol{\nabla}\xi, \boldsymbol{\nabla}\eta, \boldsymbol{\nabla}\zeta)$ are $\mathcal{O}(1/h)$, hence δ is proportional to h. Thus, in regions of appreciable δ, (5.41) reduces to an $\mathcal{O}(h)$ approximation of (5.28) for a piecewise linear finite element approximation. This behavior motivates the use of adaptive mesh refinement techniques for this class of problems since allowing $h \to 0$ in the vicinity of shock waves will increase overall solution accuracy without resorting to global mesh refinement. The adaptive mesh refinement algorithm employed will be discussed in Section 5.5.4.

Note that the combination of streamline upwinding and shock capturing required to obtain stable solutions with the finite element method is similar to the upwinding and limiting which is characteristic of total-variation-diminishing (TVD) finite difference and

finite volume schemes. TVD schemes typically employ an upwind treatment of the inviscid flux terms which is sufficient to stabilize convective-dominated flows. However, flux or slope-limiters are required in the presence of strong shock waves to restore monotonicity. Both TVD finite volume schemes and the current finite element scheme reduce to first-order at shock waves in an attempt to restore monotonicity of the solution.

For the current finite element scheme the SUPG formulation plays the same role as upwinding in finite difference and finite volume schemes. However, as in these other approaches, upwinding alone is not sufficient to produce a monotone solution. The shock capturing in the present scheme is similar to the use of limiters in that it restores monotonicity in regions of large gradients such as shock waves.

Remark: For certain classes of problems in external aerodynamics viscous effects can be neglected, yielding the Euler equations. The Euler equations are a subset of the Navier–Stokes equations which arise in the limit of vanishing viscosity. Under this limit Equation (5.28) reduces to

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_i}{\partial x_i} = 0 \tag{5.43}$$

where shock waves are now modeled as discontinuities. Now the SUPG stabilization is essential in formulating a stable method, as there is no physical viscosity to allow a non-oscillatory discretization in which $Re_c < 2$, regardless of mesh resolution. The stabilized weak statement given by Equation (5.41) becomes

$$\begin{aligned} \int_\Omega \boldsymbol{W} \cdot \left(\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}_i \frac{\partial \boldsymbol{U}}{\partial x_i} \right) \, d\Omega \\ + \sum_{e=1}^{n_{el}} \int_{\Omega_e} \boldsymbol{\tau}_{SUPG} \frac{\partial \boldsymbol{W}}{\partial x_k} \cdot \boldsymbol{A}_k \left(\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{A}_i \frac{\partial \boldsymbol{U}}{\partial x_i} \right) \, d\Omega \\ + \sum_{e=1}^{n_{el}} \int_{\Omega_e} \delta \left(\frac{\partial \boldsymbol{W}}{\partial x_i} \cdot \frac{\partial \boldsymbol{U}}{\partial x_i} \right) \, d\Omega = 0 \end{aligned} \tag{5.44}$$

The weak formulation for the Euler equations used in this work will be elaborated upon further in Section 5.3.3.3 in the context of impervious boundary condition implementation.

5.3.3 Boundary Conditions

Supersonic and hypersonic viscous and inviscid flows are considered in the subsequent numerical studies. For this class of flows the Navier-Stokes equations form a mixed elliptic-hyperbolic set of partial differential equations. Four classes of boundary conditions relevant to the problem class of interest follow:

5.3.3.1 Supersonic Inflow

At supersonic inflow boundaries the characteristics of the system are all directed into the domain, and hence each component of the system may specified as an essential boundary condition. In general, for aerothermodynamic applications the freestream density, velocity, and temperature are usually prescribed. (For aerodynamic applications the Mach number, Reynolds number, and dynamic pressure are often specified. This yields a nonlinear system which can be solved for the density, velocity, and temperature.) With these primitive variables specified the conservation variables are given directly as

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix} = \begin{bmatrix} \rho_\infty \\ \rho_\infty u_\infty \\ \rho_\infty v_\infty \\ \rho_\infty w_\infty \\ \rho_\infty \left(c_v T_\infty + \frac{\boldsymbol{u}_\infty \cdot \boldsymbol{u}_\infty}{2} \right) \end{bmatrix}$$

for the case of a calorically perfect gas.

5.3.3.2 Symmetry

Symmetry boundary conditions may be implemented for portions of the boundary which are aligned with the coordinate axes by imposing appropriate essential boundary conditions on the momentum equations. The momentum component normal to the symmetry plane is simply set equal to zero, thus constraining the resulting flow to be tangential to the boundary. As discussed in Chapter 4, assuming symmetry in the computational domain substantially reduces the size of the resulting discrete problem.

5.3.3.3 Solid Body

The Euler equations are a first–order system of partial differential equations, which is in contrast to the second–order Navier–Stokes equations. One consequence of this is that the Euler equations admit one less boundary condition at solid walls. The familiar no–slip condition for viscous flows degenerates to the no–penetration condition for the Euler condition, requiring only that the normal component of the velocity vanish on solid walls. That is,

$$\boldsymbol{u} \cdot \hat{\boldsymbol{n}} = 0 \text{ on } \Gamma_s \tag{5.45}$$

The proper way to impose this boundary condition has been discussed at length in the literature and several options have been proposed. One approach is to impose an explicit correction step in a time marching scheme to remove any normal component of velocity at no-penetration boundaries [28]. It is critical that the boundary condition be implemented in a fully implicit manner if the convergence properties of an implicit formulation are to be retained. Another approach is to transform the global coordinate axes (x, y, z) into a normal-tangential set $(\hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\eta}}, \hat{\boldsymbol{n}})$ and then impose an essential boundary condition on the normal velocity component [34, 36]. This approach has the benefit of imposing the boundary condition implicitly, but it requires the definition of a unique normal $\hat{\boldsymbol{n}}$ for nodes on the boundary. For the faceted boundary description which results from discretizing a smooth body with a mesh the normal is not defined at the nodes of elements, and produces local error in the solution, particularly at sharp corners.

In this work an alternate approach is taken in which the boundary condition is implemented through manipulation of the weak statement (5.44). To obtain the weak form of the boundary condition it is necessary to integrate the convective term in the first integral

of Equation (5.44) by parts, yielding

$$\begin{aligned}
&\int_{\Omega}\left(\boldsymbol{W}\cdot\frac{\partial\boldsymbol{U}}{\partial t}-\frac{\partial\boldsymbol{W}}{\partial x_i}\boldsymbol{A}_i\boldsymbol{U}\right)\,d\Omega\\
&+\sum_{e=1}^{n_{el}}\int_{\Omega_e}\boldsymbol{\tau}_{SUPG}\frac{\partial\boldsymbol{W}}{\partial x_k}\cdot\boldsymbol{A}_k\left(\frac{\partial\boldsymbol{U}}{\partial t}+\boldsymbol{A}_i\frac{\partial\boldsymbol{U}}{\partial x_i}\right)\,d\Omega\\
&+\sum_{e=1}^{n_{el}}\int_{\Omega_e}\delta\left(\frac{\partial\boldsymbol{W}}{\partial x_i}\cdot\frac{\partial\boldsymbol{U}}{\partial x_i}\right)\,d\Omega+\int_{\Gamma}\boldsymbol{W}\cdot\boldsymbol{f}\,d\Gamma=0
\end{aligned}\tag{5.46}$$

where the homogeneity of $\boldsymbol{F}_i(\boldsymbol{U})$ has been invoked by recognizing $\boldsymbol{F}_i(\boldsymbol{U})=\frac{\partial\boldsymbol{F}_i}{\partial\boldsymbol{U}}\boldsymbol{U}$. In (5.46) $\boldsymbol{f}=\boldsymbol{F}\cdot\hat{\boldsymbol{n}}$ is the normal component of the inviscid flux $\boldsymbol{F}$ on the boundary Γ and for three-dimensional flows is

$$\boldsymbol{F}\cdot\hat{\boldsymbol{n}}=\begin{bmatrix}\rho\boldsymbol{u}\cdot\hat{\boldsymbol{n}}\\(\rho\boldsymbol{u}\cdot\hat{\boldsymbol{n}})\,u+Pn_x\\(\rho\boldsymbol{u}\cdot\hat{\boldsymbol{n}})\,v+Pn_y\\(\rho\boldsymbol{u}\cdot\hat{\boldsymbol{n}})\,w+Pn_z\\(\rho\boldsymbol{u}\cdot\hat{\boldsymbol{n}})\,H\end{bmatrix}=\begin{bmatrix}0\\Pn_x\\Pn_y\\Pn_z\\0\end{bmatrix}\quad\text{on }\Gamma_s\tag{5.47}$$

where $\hat{\boldsymbol{n}}=n_x\hat{\boldsymbol{i}}+n_y\hat{\boldsymbol{j}}+n_z\hat{\boldsymbol{k}}$. The implicit contribution for this boundary term follows directly from invoking the homogeneity of the normal component of the inviscid flux:

$$\boldsymbol{F}\cdot\hat{\boldsymbol{n}}=\boldsymbol{A}_n\boldsymbol{U}\tag{5.48}$$

Using (5.48) in (5.46) gives

$$\begin{aligned}
&\int_{\Omega}\left(\boldsymbol{W}\cdot\frac{\partial\boldsymbol{U}}{\partial t}-\frac{\partial\boldsymbol{W}}{\partial x_i}\boldsymbol{A}_i\boldsymbol{U}\right)\,d\Omega\\
&+\sum_{e=1}^{n_{el}}\int_{\Omega_e}\boldsymbol{\tau}_{SUPG}\frac{\partial\boldsymbol{W}}{\partial x_k}\cdot\boldsymbol{A}_k\left(\frac{\partial\boldsymbol{U}}{\partial t}+\boldsymbol{A}_i\frac{\partial\boldsymbol{U}}{\partial x_i}\right)\,d\Omega\\
&+\sum_{e=1}^{n_{el}}\int_{\Omega_e}\delta\left(\frac{\partial\boldsymbol{W}}{\partial x_i}\cdot\frac{\partial\boldsymbol{U}}{\partial x_i}\right)\,d\Omega+\int_{\Gamma}\boldsymbol{W}\cdot(\boldsymbol{A}_n\boldsymbol{U})\,d\Gamma=0
\end{aligned}\tag{5.49}$$

This formulation requires the normal direction for each element residing on the no-penetration surface *on the boundary face*, which is well defined even for faceted discretizations. In numerical calculations the boundary flux is computed using the well-defined normal for each element segment coincident with the boundary.

At the surface of a body in a viscous flow the no-slip boundary condition is applied. This is implemented simply by specifying appropriate essential boundary conditions for the momentum components of the equation system. Two classes of thermal boundary conditions are also considered, adiabatic and isothermal. The adiabatic condition arises as the natural boundary condition of the weak form by omitting the boundary integral in (5.41). That is, the adiabatic condition $\boldsymbol{q} \cdot \hat{\boldsymbol{n}} = 0$ is imposed in a weak sense through the weak formulation.

The isothermal boundary condition is implemented as an essential condition on the total energy per unit volume, ρE. At a no-slip wall we have

$$\begin{aligned} \rho E &= \rho \left(e + \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2} \right) \\ &= \rho e \\ &= \rho c_v T \end{aligned}$$

which is implemented as the essential, implicit boundary condition $\rho E - \rho c_v T = 0$.

5.3.3.4 Supersonic Outflow

For inviscid supersonic flows the governing equations are hyperbolic. At outflow boundaries where the flow is supersonic all characteristics of the system are directed out of the domain. The consequence of this is that the entire outflow boundary condition is specified by the state of the fluid inside the domain. The particular weak form used to solve the Euler equations (5.46) requires that the normal momentum flux be evaluated on the outflow boundary due to the integration by parts performed on the inviscid flux term. This is implemented by simply evaluating the boundary integral in (5.49) entirely from local element contributions. That is, no specific values are assumed as all relevant information propagates outward from the interior of the domain.

At viscous supersonic outflow boundaries a similar approach is employed where the state is defined entirely by the internal conditions. However, as pointed out by Hauke and

Hughes [39], it is important to include the viscous boundary terms which result from the integration by parts performed in Equation (5.41). These boundary term contributions are computed at viscous supersonic outflow boundaries and are included in the system matrix.

5.4 Finite Element Formulation

Upon introducing a finite element discretization and corresponding basis to define the approximate solution $\boldsymbol{U}_h$ and test functions $\boldsymbol{W}_h$, and substituting into (5.41), the corresponding approximate finite element formulation has the form: Find $\boldsymbol{U}_h$ satisfying the essential boundary and initial conditions such that

$$\begin{aligned}
&\int_\Omega \left[\boldsymbol{W}_h \cdot \left(\frac{\partial \boldsymbol{U}_h}{\partial t} + \boldsymbol{A}_i \frac{\partial \boldsymbol{U}_h}{\partial x_i} \right) + \frac{\partial \boldsymbol{W}_h}{\partial x_i} \cdot \left(\boldsymbol{K}_{ij} \frac{\partial \boldsymbol{U}_h}{\partial x_j} \right) \right] \, d\Omega \\
&+ \sum_{e=1}^{n_{el}} \int_{\Omega_e} \boldsymbol{\tau}_{SUPG} \frac{\partial \boldsymbol{W}_h}{\partial x_k} \cdot \boldsymbol{A}_k \left[\frac{\partial \boldsymbol{U}_h}{\partial t} + \boldsymbol{A}_i \frac{\partial \boldsymbol{U}_h}{\partial x_i} - \frac{\partial}{\partial x_i} \left(\boldsymbol{K}_{ij} \frac{\partial \boldsymbol{U}_h}{\partial x_j} \right) \right] \, d\Omega \\
&+ \sum_{e=1}^{n_{el}} \int_{\Omega_e} \delta \left(\frac{\partial \boldsymbol{W}_h}{\partial x_i} \cdot \frac{\partial \boldsymbol{U}_h}{\partial x_i} \right) \, d\Omega - \int_\Gamma \boldsymbol{W}_h \cdot \boldsymbol{g}_h \, d\Gamma = 0
\end{aligned} \tag{5.50}$$

for all admissible test functions $\boldsymbol{W}_h$.

More specifically, let us expand $\boldsymbol{U}_h(\boldsymbol{x}, t)$ and $\boldsymbol{F}_i(\boldsymbol{x}, t)$ in terms of the finite element basis functions:

$$\boldsymbol{U}_h(\boldsymbol{x}, t) = \sum_j \phi_j(\boldsymbol{x}) \boldsymbol{U}_h(\boldsymbol{x}_j, t) \tag{5.51}$$

$$\boldsymbol{F}_i(\boldsymbol{x}, t) = \sum_j \phi_j(\boldsymbol{x}) \boldsymbol{F}_i(\boldsymbol{x}_j, t) \tag{5.52}$$

where $\boldsymbol{U}_h(\boldsymbol{x}_j, t)$ and $\boldsymbol{F}_i(\boldsymbol{x}_j, t) = \boldsymbol{A}_i \left(\boldsymbol{U}_h \left(\boldsymbol{x}_j, t \right) \right) \boldsymbol{U}_h(\boldsymbol{x}_j, t)$ are the nodal solution values and nodal inviscid flux components at time t, respectively. In this work a standard piecewise linear Lagrange basis is chosen for $\{\phi\}$, which yields a nominally second–order accurate scheme. Since the focus here is on supersonic flows which exhibit shock waves no attempt has been made to achieve higher–order spatial discretizations. (As discussed in

Section 5.3.2, the scheme is locally first–order accurate in the vicinity of shocks.) However, previous work with a similar formulation for the compressible Navier–Stokes equations suggests that the current scheme could easily be extended to higher–order for flows without shocks simply by using a higher-order finite element basis [40].

Note the particular discretization chosen in Equation (5.52) for the inviscid flux term. This approach is motivated by results which show that for the model Burger's equation this grouped discretization yields slightly higher accuracy than ungrouped scheme [41]. Recently this approach has received renewed attention in flux-corrected transport discretizations for multidimensional conservation laws [42, 43]. Applied in the current work, this approach is in contrast to previous SUPG discretizations for compressible flows (e.g. as in [34, 36, 39, 44]). To illustrate the difference consider the expansion of the steady analog to (5.52) using (5.32)

$$\begin{aligned} \boldsymbol{F}_i(\boldsymbol{x}) &= \sum_j \phi_j(\boldsymbol{x}) \boldsymbol{F}_i(\boldsymbol{x}_j) \\ &= \sum_j \phi_j(\boldsymbol{x}) \boldsymbol{A}_i\left(\boldsymbol{U}\left(\boldsymbol{x}_j\right)\right) \boldsymbol{U}(\boldsymbol{x}_j) \end{aligned} \tag{5.53}$$

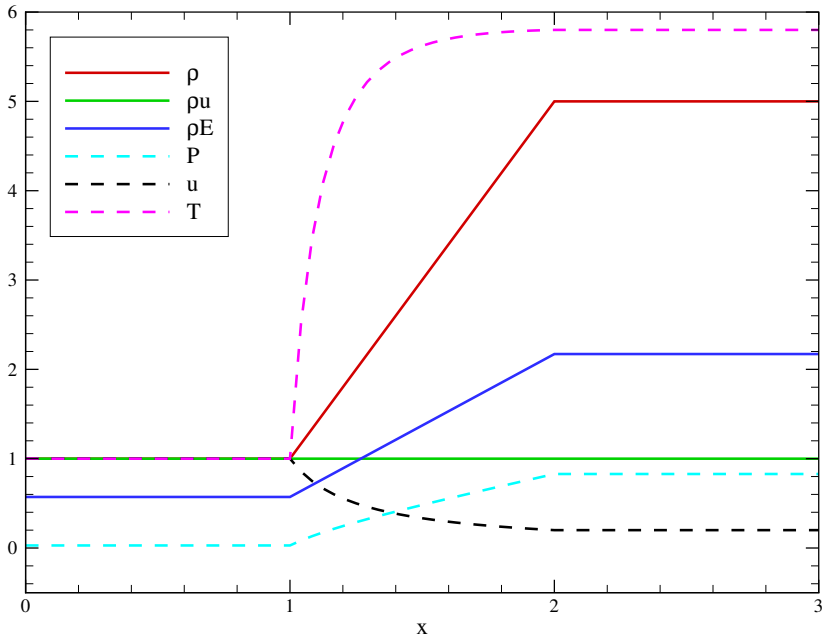
in contrast to the typical approach in which

$$\boldsymbol{F}_i(\boldsymbol{x}) = \boldsymbol{A}_i\left(\boldsymbol{U}\left(\boldsymbol{x}\right)\right) \boldsymbol{U}(\boldsymbol{x}) \tag{5.54}$$

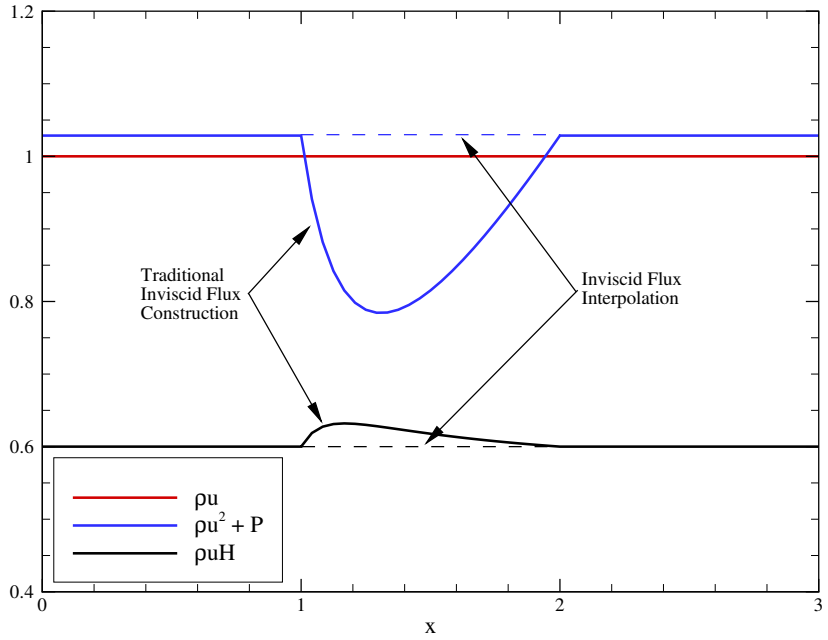
where $\boldsymbol{U}(\boldsymbol{x})$ is interpolated from nodal values as in (5.51).

Numerical experiments suggest that this choice of inviscid flux discretization improves the stability of the numerical scheme. One possible explanation for this behavior may be that in (5.53) the inviscid flux is only computed at the nodes $\boldsymbol{x}_j$ of an element and interpolated in the interior, while (5.54) evaluates the inviscid flux directly in the interior using the interpolated values $\boldsymbol{U}\left(\boldsymbol{x}\right)$.

Figure 5.1 examines why this procedure may enhance the stability of the formulation. The Figure considers a one-dimensional, inviscid, normal shock at Mach 5. For this



(a) Linearly interpolated conservation variables and reconstructed primitive variables.



(b) Linearly interpolated and reconstructed inviscid flux vector components.

Figure 5.1: A steady normal shock at Mach 5 spanning three notional elements.

simple case the governing equations reduce to

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}\left(\rho u\right) &= 0 \\
\frac{\partial \rho u}{\partial t} + \frac{\partial}{\partial x}\left(\rho u^2 + P\right) &= 0 \\
\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x}\left(\rho u H\right) &= 0
\end{aligned}$$

and, at steady state, reduce to

$$\frac{\partial}{\partial x}\left(\rho u\right) = \frac{\partial}{\partial x}\left(\rho u^2 + P\right) = \frac{\partial}{\partial x}\left(\rho u H\right) \equiv 0 \tag{5.55}$$

which implies that ρu, $\rho u^2 + P$, and $\rho u H$ are all constant.

Figure 5.1(a) presents the scenario in which the exact solution is captured on three piecewise linear finite elements of unit length. The solid lines in the figure depict the conserved variables for the nodally exact solution interpolated linearly in the finite element basis. The dashed lines are the reconstructed primitive variables, which are highly nonlinear as they are in general rational functions of the conserved variables. This is especially true in the case of the temperature, which has the form

$$T = \frac{e}{c_v} = \frac{\rho E - \frac{1}{2}\frac{(\rho u)^2}{\rho}}{\rho c_v}$$

Figure 5.1(b) plots the inviscid flux vector components for both the traditional approach and the discretization given by (5.53). Note that for the traditional approach Equation (5.55) is not satisfied within the element containing the shock, hence this scheme is incapable of representing the nodally exact solution. By contrast, for the alternate choice of (5.53) Equation (5.55) is satisfied exactly.

Recalling Equation (5.42), the inability to represent a nodally exact solution implies that the shock capturing operator will always be active in the traditional approach. The approach proposed here can satisfy the nodally exact solution and, therefore, is capable of converging to solutions in which δ vanishes throughout the domain.

Further, the distribution of $(\rho u^2 + P)$ and $\rho u H$ interior to the element containing the shock are of high-order for the traditional approach. This is important because, in practice, the integrals in the finite element weak statement (5.50) are approximated using numerical quadrature, and this high-order behavior will not be evaluated exactly.

The fact that both schemes recover identical inviscid flux values at the nodes is also important. This suggests that for a nodal quadrature rule both schemes should exhibit similar performance. This conjecture is supported by recent work in which Kessler and Awruch consider an explicit Taylor–Galerkin finite element method for the Navier–Stokes equations in thermochemical nonequilibrium [29]. In this work the authors evaluate the element integrals a priori in closed-form using Gauss-Lobatto quadrature so that at each explicit time step costly numerical integration is avoided. Given the behavior shown in Figure 5.1(b) their approach may have benefited from this enhanced stability by sampling the inviscid flux only at the element nodes.

5.5 Solution Methodology

Equations (5.50) form a transient, tightly coupled nonlinear system for the unknown nodal values $\boldsymbol{U}_h(\boldsymbol{x}_j, t)$. Even when a steady solution to the governing equations is sought equations (5.50) are often solved with a pseudo-time continuation strategy. That is, even for steady problems, the unsteady equations are often integrated in time until steady-state is reached. This is especially the case for compressible flows containing shock waves because strong gradients which occur in the flow imply an extremely small zone of attraction for nonlinear solution schemes such as Newton's method. Algorithms for solving this type of transient system fall broadly into two categories: explicit and implicit.

Explicit algorithms are relatively straightforward to implement, but are not competitive for convection-dominated problems with non-negligible viscosity because of restrictive time step sizes which must be used. Linear stability analysis for explicit schemes applied to the model convection-diffusion problem provides insight into time step stabil-

ity. For convection processes the stable time step varies linearly with the mesh spacing, that is $\Delta t_{\text{conv}} \propto h$. Analysis of the linear one-dimensional wave equation gives rise to the well-known Courant-Friedrichs-Lewy (CFL) condition, which states that for velocity u and mesh size h explicit stability requires $\frac{u\Delta t}{h} < 1$ [28]. For diffusion processes the stable time step varies *quadratically* with the mesh spacing, that is $\Delta t_{\text{diff}} \propto h^2$. For the case of an inviscid flow this condition is irrelevant and the convective time step limit may not be overly-restrictive as the minimum mesh size h may be relatively large. For convection-dominated flows with thin boundary layers, however, the extremely fine mesh spacing at solid walls which is required to accurately resolve the boundary layer renders the explicit approach intractable.

Implicit algorithms, by contrast, are substantially more expensive per time step, but have the advantage of avoiding stability limits on the size of time step used. Since the present work seeks to use adaptive meshing techniques to locally resolve fine features of the flow (thus decreasing h), the h-dependence of Δt for explicit schemes is particularly unattractive. The cost for this increased stability is the need to solve (at least approximately) a nonlinear implicit system at each time step of the solution. Preconditioned Krylov subspace iterative methods provide a suitable choice of solvers that are amenable to parallel solution and are efficient for the problems of interest here.

The remainder of this section describes (1) the time integration and (2) linearization strategies used for both steady-state and time accurate flows. The iterative techniques used to solve the resulting linear systems will also be briefly discussed.

5.5.1 Time Integration

As mentioned previously, steady solutions are often found by time-marching the transient governing equations to steady-state. In this sense the initial condition is taken at time $t = 0$ and the solution is marched in time until $\frac{\partial \boldsymbol{U}}{\partial t} \to 0$. In this way time is essentially a continuation parameter which defines a sequence $(n = 1, 2, \ldots)$ of solutions $\boldsymbol{U}_n$ which converge to the steady solution $\boldsymbol{U}$.

The semidiscrete weak form in Equation (5.50) is discretized in time using backwards finite difference schemes. Both first and second-order accurate in time schemes may be derived from Taylor series expansions in time about $\boldsymbol{U}_h\left(t_{n+1}\right) = \boldsymbol{U}_{n+1}$:

$$\begin{aligned}\boldsymbol{U}_n = \boldsymbol{U}_{n+1} &+ \frac{\partial \boldsymbol{U}_{n+1}}{\partial t}\left(t_n - t_{n+1}\right) + \frac{\partial^2 \boldsymbol{U}_{n+1}}{\partial t^2}\frac{\left(t_n - t_{n+1}\right)^2}{2} \\ &+ \mathcal{O}\left(\left(t_n - t_{n+1}\right)^3\right)\end{aligned}$$

$$\begin{aligned}\boldsymbol{U}_{n-1} = \boldsymbol{U}_{n+1} &+ \frac{\partial \boldsymbol{U}_{n+1}}{\partial t}\left(t_{n-1} - t_{n+1}\right) + \frac{\partial^2 \boldsymbol{U}_{n+1}}{\partial t^2}\frac{\left(t_{n-1} - t_{n+1}\right)^2}{2} \\ &+ \mathcal{O}\left(\left(t_{n-1} - t_{n+1}\right)^3\right)\end{aligned}$$

which, upon substituting $t_{n+1} - t_n \equiv \Delta t_{n+1}$ and $t_{n+1} - t_{n-1} = \Delta t_{n+1} + \Delta t_n$, becomes

$$\boldsymbol{U}_n = \boldsymbol{U}_{n+1} - \frac{\partial \boldsymbol{U}_{n+1}}{\partial t}\Delta t_{n+1} + \frac{\partial^2 \boldsymbol{U}_{n+1}}{\partial t^2}\frac{\Delta t_{n+1}^2}{2} - \mathcal{O}\left(\Delta t_{n+1}^3\right)$$

$$\begin{aligned}\boldsymbol{U}_{n-1} = \boldsymbol{U}_{n+1} &- \frac{\partial \boldsymbol{U}_{n+1}}{\partial t}\left(\Delta t_{n+1} + \Delta t_n\right) + \frac{\partial^2 \boldsymbol{U}_{n+1}}{\partial t^2}\frac{\left(\Delta t_{n+1} + \Delta t_n\right)^2}{2} \\ &- \mathcal{O}\left(\left(\Delta t_{n+1} + \Delta t_n\right)^3\right)\end{aligned}$$

which can be rewritten for $\frac{\partial \boldsymbol{U}_{n+1}}{\partial t}$ as:

$$\frac{\partial \boldsymbol{U}_{n+1}}{\partial t} = \frac{\boldsymbol{U}_{n+1}}{\Delta t_{n+1}} - \frac{\boldsymbol{U}_n}{\Delta t_{n+1}} + \frac{\partial^2 \boldsymbol{U}_{n+1}}{\partial t^2}\frac{\Delta t_{n+1}}{2} - \mathcal{O}\left(\Delta t_{n+1}^2\right) \tag{5.56}$$

$$\begin{aligned}\frac{\partial \boldsymbol{U}_{n+1}}{\partial t} = &\frac{\boldsymbol{U}_{n+1}}{\Delta t_{n+1} + \Delta t_n} - \frac{\boldsymbol{U}_{n-1}}{\Delta t_{n+1} + \Delta t_n} + \frac{\partial^2 \boldsymbol{U}_{n+1}}{\partial t^2}\frac{\left(\Delta t_{n+1} + \Delta t_n\right)}{2} \\ &- \mathcal{O}\left(\left(\Delta t_{n+1} + \Delta t_n\right)^2\right)\end{aligned} \tag{5.57}$$

5.5.1.1 Time Marching to Steady-State

The familiar backward Euler time discretization follows directly from (5.56) by recognizing

$$\frac{\partial \boldsymbol{U}_{n+1}}{\partial t} = \frac{\boldsymbol{U}_{n+1}}{\Delta t_{n+1}} - \frac{\boldsymbol{U}_n}{\Delta t_{n+1}} + \mathcal{O}\left(\Delta t_{n+1}\right) \tag{5.58}$$

which provides a first-order in time approximation upon neglecting the $\mathcal{O}\left(\Delta t_{n+1}\right)$ term. As such, this scheme yields a fully implicit problem for $\boldsymbol{U}_{n+1}$ which may be used when time accuracy is not required.

5.5.1.2 Time-Accurate Flows

A linear combination of (5.56) scaled by $\left(1+\frac{\Delta t_{n+1}}{\Delta t_n}\right)$ and (5.57) scaled by $-\frac{\Delta t_{n+1}}{\Delta t_n}$ can be used to annihilate the leading $\frac{\partial^2 \boldsymbol{U}_{n+1}}{\partial t^2}$ term and create a backward, second-order accurate approximation to $\frac{\partial \boldsymbol{U}_{n+1}}{\partial t}$. This approximation, along with (5.58), can be generalized in the form

$$\frac{\partial \boldsymbol{U}_{n+1}}{\partial t} = \alpha_t \boldsymbol{U}_{n+1} + \beta_t \boldsymbol{U}_n + \gamma_t \boldsymbol{U}_{n-1} + \mathcal{O}\left(\Delta t_{n+1}^p\right) \tag{5.59}$$

to yield either a first or second-order accurate scheme. The weights α_t, β_t, and γ_t are given for $p = 1$ and $p = 2$ in Table 5.1. Since this second-order scheme requires *two* levels

Table 5.1: First and second-order accurate time discretization coefficients.

$\boldsymbol{p}$	$\boldsymbol{\alpha_t}$	$\boldsymbol{\beta_t}$	$\boldsymbol{\gamma_t}$
1	$\frac{1}{\Delta t_{n+1}}$	$\frac{-1}{\Delta t_{n+1}}$	0
2	$-\beta_t - \gamma_t$	$-\left[\frac{1}{\Delta t_{n+1}} + \frac{1}{\Delta t_n}\right]$	$\frac{\Delta t_{n+1}}{\Delta t_n(\Delta t_{n+1}+\Delta t_n)}$

of solution history it is not self-starting. In practice ten backwards Euler steps are taken to develop the required solution history and to allow rapid transients to subside before applying the second-order scheme.

5.5.2 Linearization

After time discretization using either (5.58) or (5.59), Equation (5.50) can be written in residual form for the unknown nodal values $\boldsymbol{U}_{n+1} \equiv \boldsymbol{U}_h\left(t_{n+1}\right)$ as the nonlinear

algebraic system

$$\boldsymbol{R}\left(\boldsymbol{U}_{n+1}\right) = 0 \tag{5.60}$$

The goal of this section is to define a sequence of linear problems that, when solved, converge to obtain the solution $\boldsymbol{U}_{n+1}$ of the nonlinear system (5.60).

5.5.2.1 Newton Scheme

Expanding (5.60) with a Taylor series about iterate $\boldsymbol{U}^{l}_{n+1}$ gives

$$\boldsymbol{R}\left(\boldsymbol{U}^{l+1}_{n+1}\right) = \boldsymbol{R}\left(\boldsymbol{U}^{l}_{n+1}\right) + \left[\frac{\partial \boldsymbol{R}\left(\boldsymbol{U}^{l}_{n+1}\right)}{\partial \boldsymbol{U}_{n+1}}\right] \delta\boldsymbol{U}^{l+1}_{n+1} + \mathcal{O}\left(\left(\delta\boldsymbol{U}^{l+1}_{n+1}\right)^{2}\right) \tag{5.61}$$

where $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}$ is the Jacobian matrix for the nonlinear system and $\delta\boldsymbol{U}^{l+1}_{n+1} = \boldsymbol{U}^{l+1}_{n+1} - \boldsymbol{U}^{l}_{n+1}$. Truncating this expansion and setting $\boldsymbol{R}\left(\boldsymbol{U}^{l+1}_{n+1}\right) = 0$ yields Newton's method

$$\begin{aligned} 0 &= \boldsymbol{R}\left(\boldsymbol{U}^{l}_{n+1}\right) + \left[\frac{\partial \boldsymbol{R}\left(\boldsymbol{U}^{l}_{n+1}\right)}{\partial \boldsymbol{U}_{n+1}}\right] \delta\boldsymbol{U}^{l+1}_{n+1} \\ \left[\frac{\partial \boldsymbol{R}\left(\boldsymbol{U}^{l}_{n+1}\right)}{\partial \boldsymbol{U}_{n+1}}\right] \delta\boldsymbol{U}^{l+1}_{n+1} &= -\boldsymbol{R}\left(\boldsymbol{U}^{l}_{n+1}\right) \end{aligned} \tag{5.62}$$

which results in an implicit linear system for $\delta\boldsymbol{U}^{l+1}_{n+1}$ and a sequence of iterates ($l = 0, 1, \ldots$) which converges to $\boldsymbol{U}_{n+1}$. It is important to recall than Newton's method exhibits second-order *conditional* convergence. That is, the magnitude of $\boldsymbol{R}\left(\boldsymbol{U}^{l+1}_{n+1}\right)$ decreases quadratically at successive iterates provided that the initial guess $\boldsymbol{U}^{0}_{n+1}$ is "sufficiently close" to the unknown $\boldsymbol{U}_{n+1}$ [20, 45].

While the full-Newton scheme is conceptually simple the implementation is complicated by the nonlinear dependence of the transport properties on the unknowns (see Equation (5.17)) and the highly nonlinear nature of the convective terms themselves. In practice, implementing the full-Newton scheme is computationally intensive and, in the case of supersonic flows exhibiting shock waves, is often only of modest benefit. That is, due to the conditional convergence restriction of the method and the sharp gradients

or discontinuities which are present in the flowfield, the asymptotic quadratic convergence rate may not be achieved [46]. The implementation of an approximate Newton-Krylov technique to address these issues will be discussed further in Section 5.5.3.2.

5.5.2.2 Frozen Coefficient Scheme

The frozen coefficient scheme form results when the coefficient matrices in (5.50) are evaluated using the last solution in a successive approximation nonlinear iteration. This approach allows the linear system to be assembled easily at each nonlinear step and is computationally attractive.

However, numerical experiments show erratic convergence for this approach. For example, nonlinear convergence stagnates and the magnitude of the time derivative $\|\frac{\partial \boldsymbol{U}}{\partial t}\|$ stops decreasing at some finite value. Shakib et al. [46] point out that the frozen coefficient scheme does not necessarily yield a descent direction for $\boldsymbol{R}(\boldsymbol{U})$. That is, each successive iterate is not guaranteed to produce a solution which reduces the residual. Bearing this in mind, this convergence stagnation is not surprising. It is simply the price to pay for considering an incomplete Jacobian approximation of (5.61). Similar convergence stagnation has been observed in the finite volume community when particular flux limiters are used.

A practical solution to this problem has recently been proposed by Catabriga and Coutinho [44]. The solution proposed by Catabriga and Coutinho in the context of steady flows is to "freeze" the shock capturing parameter at some point in the computation. The nonlinear solution then proceeds with δ fixed at some previous value and the scheme will typically converge to machine precision.

5.5.3 Linear System Solution Scheme

Both the Newton and frozen coefficient Jacobian schemes result in a series of sparse linear problems of the form

$$\boldsymbol{K}\ \delta \boldsymbol{U}_{n+1} = \boldsymbol{f} \tag{5.63}$$

which must be solved to obtain $\boldsymbol{U}_{n+1}$. For the discretization presented in Section 5.4 using standard piecewise-linear elements $\boldsymbol{K}$ is a non-symmetric, sparse, non-singular $(nv \times n_{\text{nodes}}) \times (nv \times n_{\text{nodes}})$ real matrix where n_{nodes} is the total number of nodes in the mesh and nv is the number of unknowns per node (which is equal to four in two dimensions and five in three dimensions). Given the size and sparseness of $\boldsymbol{K}$ it is natural to use preconditioned Krylov subspace iterative techniques to approximate $\delta\boldsymbol{U}_{n+1}$ [47, 48]. The essential kernel of these techniques is the computation of the matrix-vector product $\boldsymbol{y} = \boldsymbol{K}\ \boldsymbol{x}$. Two techniques for providing this kernel will be discussed, the first stores the sparse matrix and computes the matrix-vector product explicitly; the second computes the action of the matrix-vector product in a "matrix-free" sense.

5.5.3.1 Sparse Matrix Approach

One straightforward technique for solving (5.63) is to build the system matrix $\boldsymbol{K}$ and right-hand-side vector $\boldsymbol{f}$. Since the matrix is large yet sparse care must be taken to store it efficiently. In the present work the parallel sparse matrix format implemented in the PETSc toolkit is used, as are the PETSc iterative solvers [15]. When the system matrix is constructed explicitly it may then be copied and modified to serve as a preconditioner as well. (This is the approach used in this work.) In the current work a standard parallel block-Jacobi ILU-0 preconditioner is used [47, 48]. Once the system matrix and preconditioner are formed the required matrix-vector products are computed directly.

5.5.3.2 Matrix-Free Approach

Recall from Equation (5.62) the particular form of the implicit system to be solved:

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]\ \delta\boldsymbol{U} = -\boldsymbol{R}\left(\boldsymbol{U}\right)$$

For this special case the action of the matrix-vector product $\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]\ \delta\boldsymbol{U}$ is nothing more than the derivative of $\boldsymbol{R}$ in the direction specified by $\delta\boldsymbol{U}$, which is formally defined as

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]\ \delta\boldsymbol{U} \equiv \lim_{\varepsilon \to 0}\left\{\frac{\boldsymbol{R}\left(\boldsymbol{U} + \varepsilon\delta\boldsymbol{U}\right) - \boldsymbol{R}\left(\boldsymbol{U}\right)}{\varepsilon}\right\}$$

and may be approximated within $\mathcal{O}(\varepsilon)$ for finite ε as

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right] \delta \boldsymbol{U} \approx \frac{\boldsymbol{R}(\boldsymbol{U} + \varepsilon \delta \boldsymbol{U}) - \boldsymbol{R}(\boldsymbol{U})}{\varepsilon} \tag{5.64}$$

From Equation (5.64) it is clear that the required matrix-vector product may be approximated by differencing successive residual evaluations. It is in this sense that the scheme is matrix-free: the actual system matrix need not be explicitly formed. All that is required is the capability to evaluate the discrete residual $\boldsymbol{R}(\boldsymbol{U})$. Of course, for practical applications some form of preconditioning must be applied to the linear system. Depending on the implementation of this preconditioning the composite scheme may store some approximation of the system matrix. Still, one attractive feature of the matrix-free approach is that it can require substantially less memory than the sparse matrix approach.

Perhaps the most compelling reason to use the matrix-free approach is that it directly yields a quasi-Newton formulation. That is, the finite difference approximation properly accounts for *all* the nonlinearities in the system. This is especially attractive from an algorithm development perspective. For example, alternate shock capturing terms, SUPG weighting functions, equations of state, and transport property definitions can all be implemented simply by defining their contribution to the discrete residual. Their contribution to the quasi-Newton iteration simply falls out through the approximate matrix-vector product (5.64).

The choice of ε is critical for the success of the method and has been the subject of much research and will not be discussed in detail. The primary concern is achieving ε sufficiently small that (5.64) is reasonably accurate while avoiding catastrophic cancellation which may occur when evaluating the numerator with finite precision arithmetic [46]. However, it is worth mentioning that an alternate approach for approximating the directional derivative is available (using complex arithmetic) which is second-order accurate in ε, poses no risk for cancellation errors, and is of similar cost . A simple Taylor series for a real valued function $f(x)$ evaluated for the complex perturbation $(x + i\,\varepsilon)$ gives

$$\frac{df}{dx} = \frac{\Im\left[f(x + i\,\varepsilon)\right]}{\varepsilon} + \mathcal{O}\left(\varepsilon^2\right)$$

where $\Im[]$ denotes the imaginary part of a complex number. This robust technique has recently been applied by Nielsen and Kleb as a method for computing discrete adjoint operators for a finite volume formulation of the Navier-Stokes equations on unstructured meshes [49]. Future work will consider this technique in the context of matrix-free Newton-Krylov methods as an alternate approach for constructing an approximate Jacobian matrix.

5.5.4 Adaptive Mesh Refinement

The adaptive solution techniques discussed in the previous chapters may be applied to great effect for the case of supersonic compressible flows. As mentioned in the introductory remarks of this chapter, this class of flows is characterized by highly localized features. Shock waves which may occur are physically on the order of several molecular mean-free paths wide (on the order of micrometers for air at sea level conditions). Compared to the scale of flight vehicles, these shock waves appear essentially as discontinuities. Further, their precise location is not known a priori, so it is natural to use an adaptive technique to accurately capture these features.

The boundary layer adjacent to solid surfaces in viscous flows is another region of very high gradients which occur in a small region of the domain. Typical practice for aerodynamic and aerothermodynamic applications is to construct a computational mesh which is expected to capture these features accurately. Viscous shear layers pose a similar but more complex problem in that their precise location is not known during the mesh generation process. For cases involving complicated and/or transient features such as shock wave/boundary layer interaction, this mesh generation technique necessarily becomes an iterative one.

Adaptive mesh refinement allows an alternative to this cumbersome iterative solution procedure by embedding the mesh improvement process directly in the application code. In the numerical examples which follow these adaptive techniques will be used to provide local resolution for strong shock waves which occur in both viscous and inviscid flows. Additionally, the technique is applied to viscous/inviscid interactions and complex

shock-shock interactions to assess their viability for performing aerothermodynamic applications.

The solution procedure with adaptive mesh refinement is largely the same as described in Chapter 4. One significant departure here is that in some cases 'feature indicators' are used as an alternative means for selecting which portions of the domain are to be refined. This feature-based approach does not enjoy the rigorous theoretical support of true error estimation techniques, but has worked well in practice for a number of applications (as will be demonstrated).

5.5.5 Solution Algorithm

The solution methodologies described in the previous sections are combined into the transient adaptive nonlinear solution algorithm listed as Algorithm 5.1. This is the algorithm that is used to perform the application studies presented in the next section.

Algorithm 5.1 Transient adaptive nonlinear solution algorithm used for compressible flow applications

Interpolate initial conditions
$\boldsymbol{U}_0 = \boldsymbol{U}(\boldsymbol{x}, t)$
for $n = 1$ to $N_{\text{time steps}}$ **do**
  Let $\hat{\boldsymbol{U}}_n = \boldsymbol{U}_{n-1}$
  Solve the nonlinear system for $\hat{\boldsymbol{U}}_n$:
  do
    Form system matrix $\boldsymbol{K} = \boldsymbol{K}(\hat{\boldsymbol{U}}_n)$
    Form system vector $\boldsymbol{f} = \boldsymbol{f}(\hat{\boldsymbol{U}}_n, \boldsymbol{U}_{n-1}, \boldsymbol{U}_{n-2})$
    Solve the linear system $\boldsymbol{K}\ \delta\hat{\boldsymbol{U}}_n = \boldsymbol{f}$
    Update the solution $\hat{\boldsymbol{U}}_n \leftarrow \hat{\boldsymbol{U}}_n + \delta\hat{\boldsymbol{U}}_n$
  while $\left\|\delta\hat{\boldsymbol{U}}_n\right\|_\infty < \varepsilon_{nl}$
  Compute error indicator for each element using $\hat{\boldsymbol{U}}_n$
  if error is acceptable
    Let $\boldsymbol{U}_n = \hat{\boldsymbol{U}}_n$
  else
    Refine and coarsen mesh
    Project $\Pi\hat{\boldsymbol{U}}_n \rightarrow \boldsymbol{U}_n$
    Solve the nonlinear system for $\boldsymbol{U}_n$:
    do
      Form system matrix $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{U}_n)$
      Form system vector $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{U}_n, \boldsymbol{U}_{n-1}, \boldsymbol{U}_{n-2})$
      Solve the linear system $\boldsymbol{K}\ \delta\boldsymbol{U}_n = \boldsymbol{f}$
      Update the solution $\boldsymbol{U}_n \leftarrow \boldsymbol{U}_n + \delta\boldsymbol{U}_n$
    while $\|\delta\boldsymbol{U}_n\|_\infty < \varepsilon_{nl}$
  endif
end

5.6 Application Studies

This section presents a number of application studies designed to test various aspects of the finite element algorithm described in Section 5.4 which is used to solve equations (5.1)–(5.3). The application code used for these studies is built on top of the `libMesh`[1] parallel adaptive finite element library and uses a fully implicit scheme to solve the weak formulation presented in (5.41). The problems considered here are generally hypersonic, laminar, perfect gas flows in two and three dimensions.

The applications are run on computational resources ranging from desktop machines to workgroup-class parallel clusters to the *Columbia* supercomputer. All computations employ the PETSc toolkit from Argonne National Laboratory [15] to solve the parallel implicit linear systems using the generalized minimum residual (GMRES) Krylov subspace technique [50] with preconditioning. The preconditioner is of parallel block Jacobi-type where each processor sub-block uses an overlapping additive Schwartz method with an incomplete lower-upper factorization at the sub-block level with no fill (ILU-0). Spatial integration is performed with Gauss quadrature rules sufficient to integrate 3^{rd}–order polynomials exactly.

The first applications will focus on the numerical aspects of the present finite element algorithm, investigating issues such as accuracy, efficiency, and convergence. More difficult applications are then selected for phenomenological study and to demonstrate the broad applicability of the scheme, particularly to the case of steady and unsteady hypersonic aerothermodynamics.

[1] `http://libmesh.sourceforge.net`

5.6.1 Inviscid Flow over a Cylinder

5.6.1.1 Geometry and Flow Conditions

Two-dimensional inviscid Mach 3 flow over a circular cylinder is an established benchmark problem [51, 52] and is studied here to investigate the performance of the implicit formulation. The exterior flow problem is posed on a finite subdomain with uniform far-field data prescribed on the inflow boundary. The computational grid for this case is mapped from the unit square $[0,1] \times [0,1]$ in the (ξ, η) plane by

$$x(\xi,\eta) = (R_x - (R_x - R_c)\,\xi)\cos(\theta\,(2\eta - 1)) \tag{5.65}$$

$$y(\xi,\eta) = (R_y - (R_y - R_c)\,\xi)\sin(\theta\,(2\eta - 1)) \tag{5.66}$$

where the cylinder radius $R_c = 0.5$, the upstream boundary of the computational domain is given by $R_x = 1.5$, $R_y = 3$, and $\theta = \frac{5\pi}{12}$. A coarse mesh is shown in Figure 5.2 with $n_\xi \times n_\eta = 30 \times 40$ elements in the normal and circumferential directions, respectively. The particular form of this mapping has been used in high-order finite difference discretizations as it yields a smooth, differentiable mapping from computational to physical space [52].

The simulation is initialized with uniform freestream values and marched in time until steady–state is reached. The upstream inflow boundary uses a supersonic boundary condition in which the conserved variables $[\rho, \rho u, \rho v, \rho E]^T$ are specified as essential boundary conditions. The downstream outflow for this case is supersonic, and hence no outflow boundary conditions are specified for this inviscid flow. The no–penetration boundary condition $\boldsymbol{u} \cdot \hat{\boldsymbol{n}} = 0$ holds on the cylinder surface and is enforced as a natural boundary condition through the boundary integral in the weak statement as described in Section 5.3.3.

This example will be treated as a prototype for convection-dominated supersonic and hypersonic problems which arise in aerospace engineering. The performance of the finite element algorithm presented in the previous sections will be examined in detail for this example. The results of the numerical experiments performed in this section will be generalized and applied to more physically complicated flow phenomena in later sections.

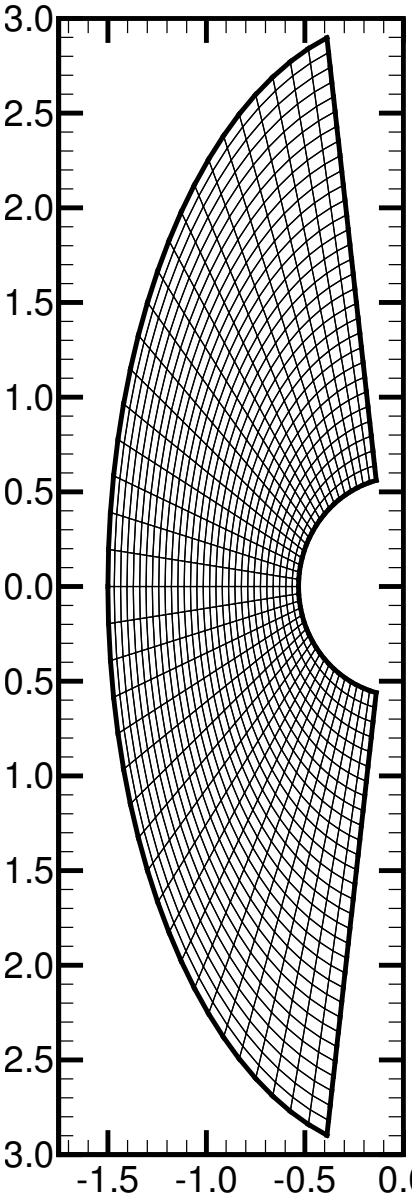


Figure 5.2: Coarse computational grid for Mach 3 flow over a cylinder

5.6.1.2 Flowfield and Stagnation Line Properties

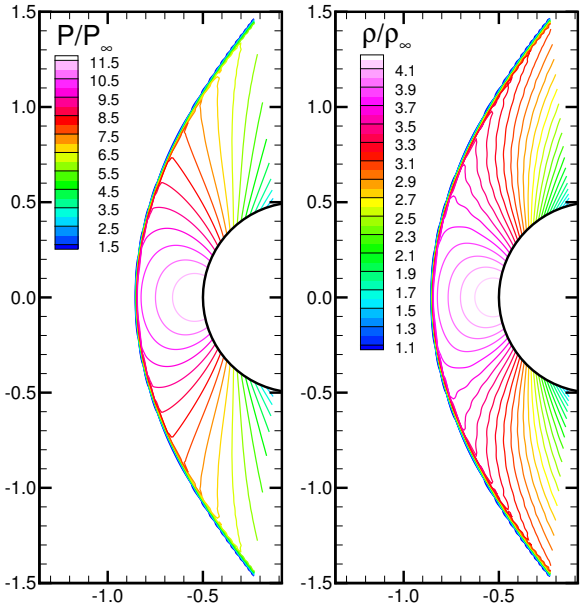
Figure 5.3 illustrates the steady-state flowfield for this case. For this inviscid case the governing Euler equations (5.43) are hyperbolic and admit discontinuous solutions. As expected, the cylinder produces a strong bow shock across which the density, velocity, and pressure jump, as predicted by the Rankine–Hugoniot equations (see Appendix A.3).

Figure 5.4 shows the flowfield properties along the stagnation line versus nondimensional distance x/R_N. It is apparent from the figure that the bow shock is located at approximately $0.7\,R_N$ upstream from the stagnation point, which agrees well with experimentally measured values [53]. As expected, the pressure, density, and temperature all increase across the shock wave while the Mach number decreases. The computed jumps are in excellent agreement with theoretical predictions as evident in Table 5.2. This indicates that the numerical scheme is properly reproducing the shock jump conditions, which is expected for any viable formulation based on the conservation form of the governing equations (5.1)–(5.3). Note that other formulations which are not based on the conservation form of the governing equations are possible and may provide simpler discretizations [54]. In general, however, these formulations *will not* converge to a solution which satisfies the Rankine–Hugoniot equations. Therefore, such schemes would not produce proper jump conditions for this case.

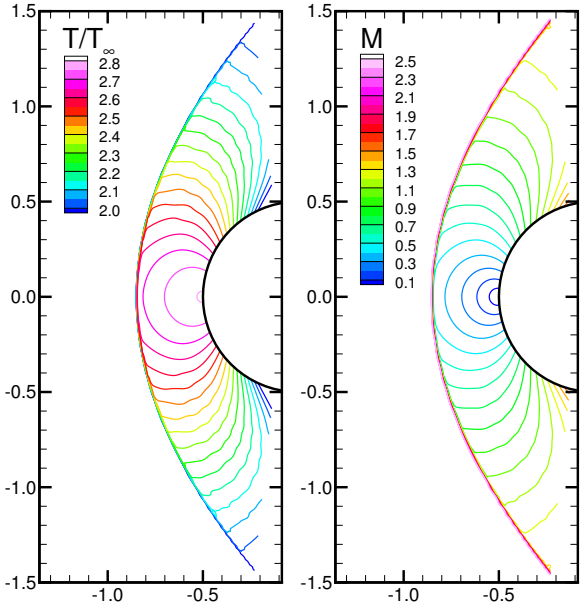
Table 5.2: Computed and theoretical jump values for a Mach 3 normal shock.

	P/P_∞	ρ/ρ_∞	T/T_∞
Theory [55]	10.33	3.857	2.679
Computation	10.34	3.854	2.683

This example also serves as a good test case for the shock–capturing operator δ. The stagnation line profiles depicted in Figure 5.4 show that the shock is captured over 2–3 elements without spurious oscillations when the shock is essentially aligned with the grid.



(a) Pressure (b) Density



(c) Temperature (d) Mach Number

Figure 5.3: Illustration of flowfield for Mach 3 flow over a cylinder

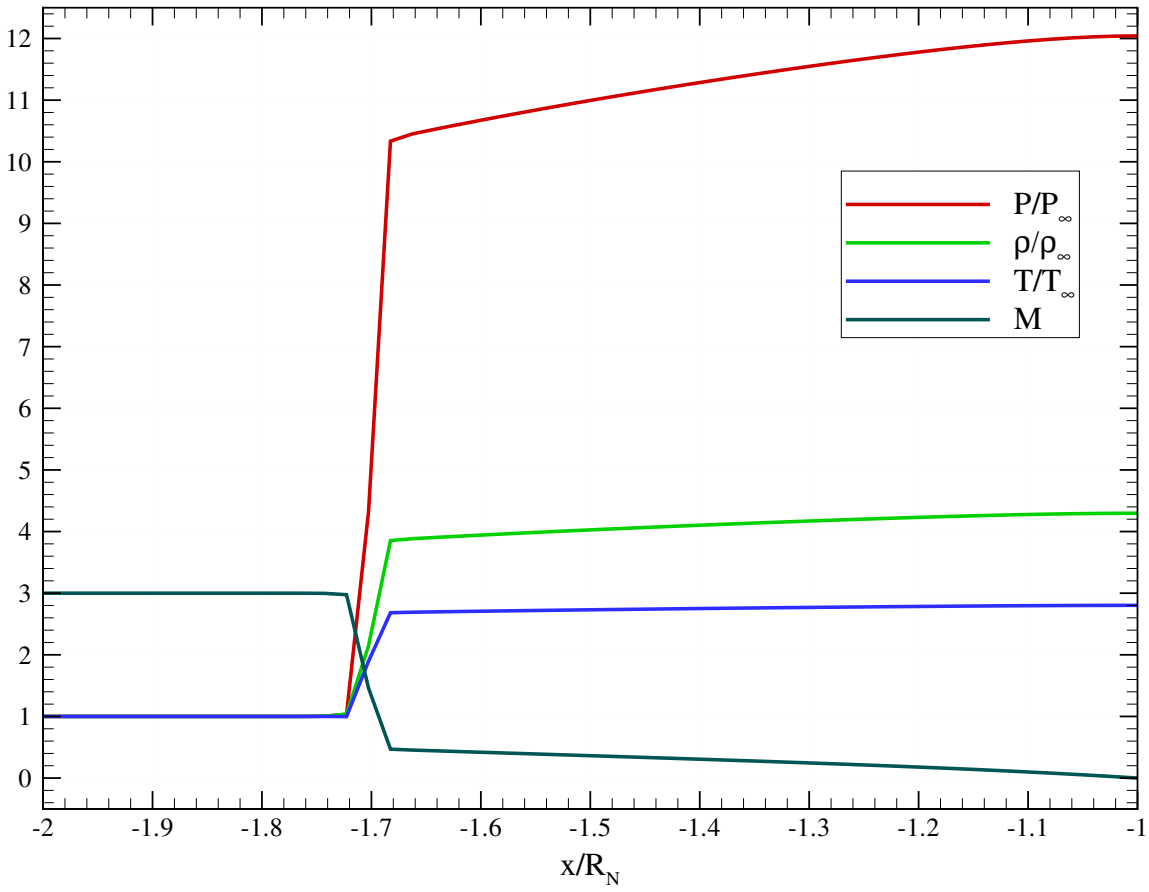


Figure 5.4: Stagnation line profile for Mach 3 flow over a cylinder

This stagnation line will be revisited in Section 5.6.1.4 in the context of mesh convergence and adaptive mesh refinement.

5.6.1.3 Convergence

In order to better characterize both the transient and nonlinear discretization schemes described in Section 5.5, a series of numerical experiments were conducted which varied (1) the order of the time discretization and (2) the number of subiterations used to solve the discrete, nonlinear, implicit problem which results at each time step. For these higher fidelity numerical experiments a mesh of $n_\xi \times n_\eta = 120 \times 160$ elements was used. A discussion of mesh convergence will be presented following the algorithmic performance investigation.

5.6.1.3.1 Temporal Convergence to Steady-State The absence of viscosity-induced separation, wake flow, and shock interaction produces a fairly simple, steady flowfield. It is therefore expected that the numerical scheme will converge to a steady-state, which is assumed to be reached when the discrete unsteady residual, $\frac{\Delta \boldsymbol{U}}{\Delta t}$, falls below a user-specified tolerance ε_{ss} in the maximum norm. That is, steady-state is assumed when

$$\mathcal{R}_n \equiv \left\| \frac{\Delta \boldsymbol{U}_n}{\Delta t_n} \right\|_\infty < \varepsilon_{ss} \tag{5.67}$$

where ε_{ss} is the steady-state solution tolerance and was taken as $\varepsilon_{ss} = 10^{-12}$. The simulation begins with the domain initialized to freestream conditions everywhere and a user-specified initial time step Δt_0 is used to advance the solution, which was taken here as 2×10^{-3}. The time step is allowed to grow geometrically with the relative change in the unsteady residual measured over k time steps. Explicitly,

$$\Delta \bar{t}_{n+1} = \Delta t_{n-k} \left[\frac{\mathcal{R}_{n-k}}{\mathcal{R}_n} \right]^r \tag{5.68}$$

$$\Delta t_{n+1} = \min \left(\Delta \bar{t}_{n+1}, \Delta t_{\max} \right) \tag{5.69}$$

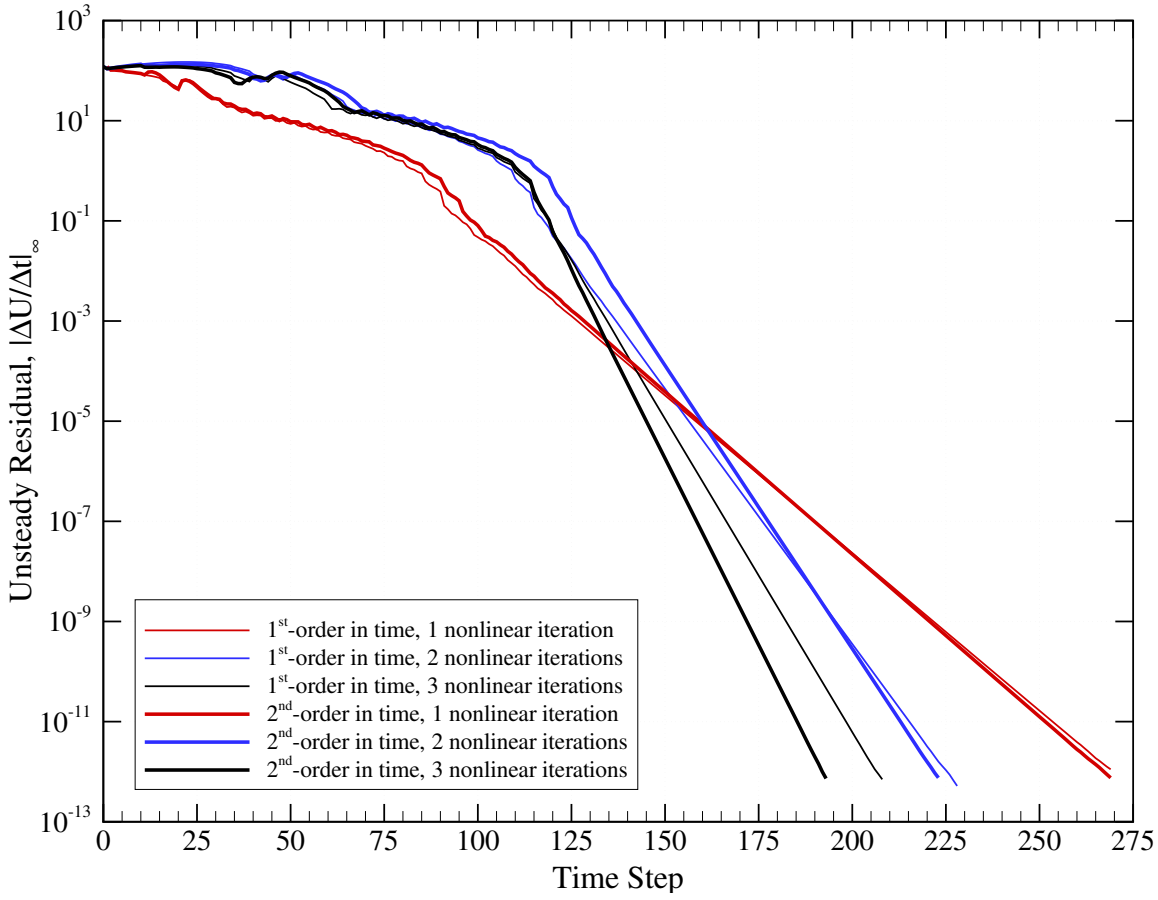


Figure 5.5: Time step convergence history for Mach 3 flow over a cylinder for a range of nonlinear solver subiterations and time discretizations.

where r is the geometric time step growth rate, which was fixed at 1.2 in this case. The time step size is updated every $k = 5$ time steps and the maximum allowable time step $\Delta t_{\text{max}} = 1$ corresponds to the amount of time required for a fictitious point in the freestream to be convected one cylinder diameter.

One immediate observation from the numerical experiments is that the 1st- and 2nd-order time discretizations exhibit similar transient convergence behavior. The convergence history exhibits two distinct phases: (1) the pre-asymptotic phase in which the bow shock develops and travels upstream to its steady location and (2) the asymptotic phase where large-scale changes in the flowfield have subsided and the remaining transient behavior is damped out.

In the pre-asymptotic phase the time discretization order of accuracy has little influence on the convergence rate. This is consistent with the observation that, during this highly nonlinear process, the time step size must be limited to achieve convergence of the nonlinear subproblem. In the asymptotic phase the convergence rate of the two schemes is again comparable. Since the only added cost associated with the 2nd-order scheme is the storage of an additional solution vector, there seems to be no compelling motivation to use the 1st-order scheme. Additionally, using the 2nd-order scheme will more accurately capture any unsteady flow phenomena which might occur for a given configuration.

It is interesting that the current finite element scheme does not exhibit the nonlinear residual convergence stagnation noted by Catabriga and Coutinho when using a very similar SUPG finite element scheme for the conservation variables [44]. This difference must be due to either (i) the inviscid flux treatment used in the present scheme or (ii) the integration by parts performed on the inviscid flux terms since the remainder of the discretized weak form is identical.

5.6.1.3.2 Nonlinear Solver Accuracy A first glance at Figure 5.5 might suggest that the algorithm performs better when specifying a larger number of nonlinear iterations per time step. In this case the figure is misleading because, in the current implementation, the

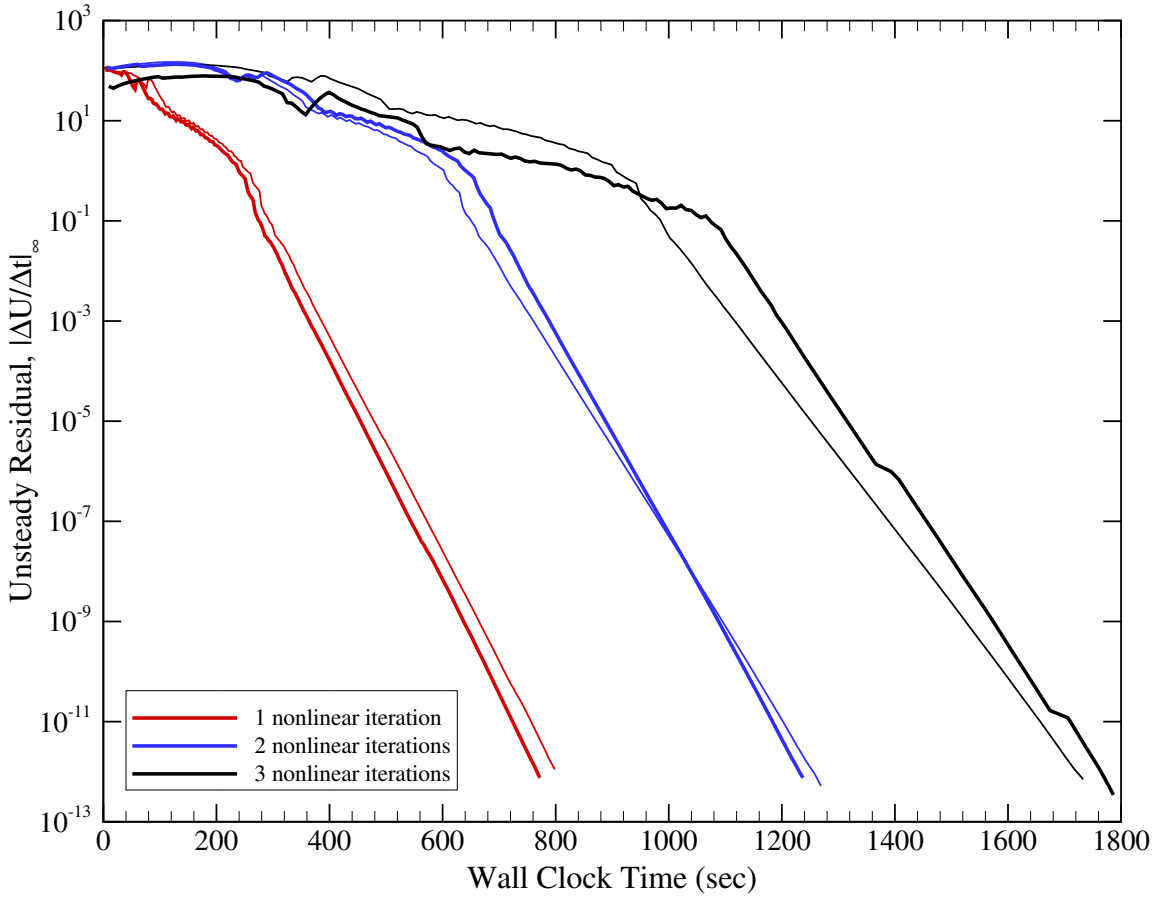


Figure 5.6: Wall clock convergence history for Mach 3 flow over a cylinder for a range of nonlinear solver subiterations and time discretizations.

computational cost of each time step is proportional to the number of nonlinear iterations used. Figure 5.6 shows the unsteady residual versus wall-clock time for the cases previously mentioned. Note that the trend observed in the previous figure is now reversed. The wall clock time is seen to increase directly with the number of nonlinear iterates. Thus, even though the case of three nonlinear iterations per time step converges in the shortest number of physical time steps it clearly is the most expensive in terms of wall clock time.

This study supports the common practice of performing only one nonlinear solution iterate per time step when considering steady flows, therefore future examples which consider stationary flows will only perform one nonlinear iteration per time step. It must be emphasized that this truncated nonlinear problem is essentially a pseudo-time continuation procedure and, that for cases where transient behavior is of interest, the nonlinear problem at each time step must be solved to an accuracy commensurate with other aspects of the algorithm.

While the wall clock time required increases with the number of nonlinear solver subiterations, it is interesting that it does not increase linearly. This is because at each subsequent nonlinear iteration the underlying linear system is solved with an iterative Krylov subspace method which benefits from the accuracy of the initial iterate. The linear solver for each subsequent nonlinear iteration in general converges more rapidly than the one before, hence the overall scaling is not linear with the number of nonlinear subiterations.

Note that in this work the preconditioning matrix used in the linear solver is assembled and factored for each linear solve. Follow-on work could consider the performance trade-off between recomputing the preconditioning matrix versus fixing the preconditioner for some number of iterations and accepting a less accurate approximate inverse matrix. Similar techniques were investigated by Barth [56] for incompressible non-Newtonian fluids and show promise for reducing the computational effort required per time step.

5.6.1.3.3 Mesh Convergence A series of nested meshes consisting of $n_\xi \times n_\eta = 60 \times 80$, 120×160, and 240×320 elements was used, as well as an adaptive mesh.Typical results

for the stagnation line density profile are shown in Figure 5.7. The figure shows the density

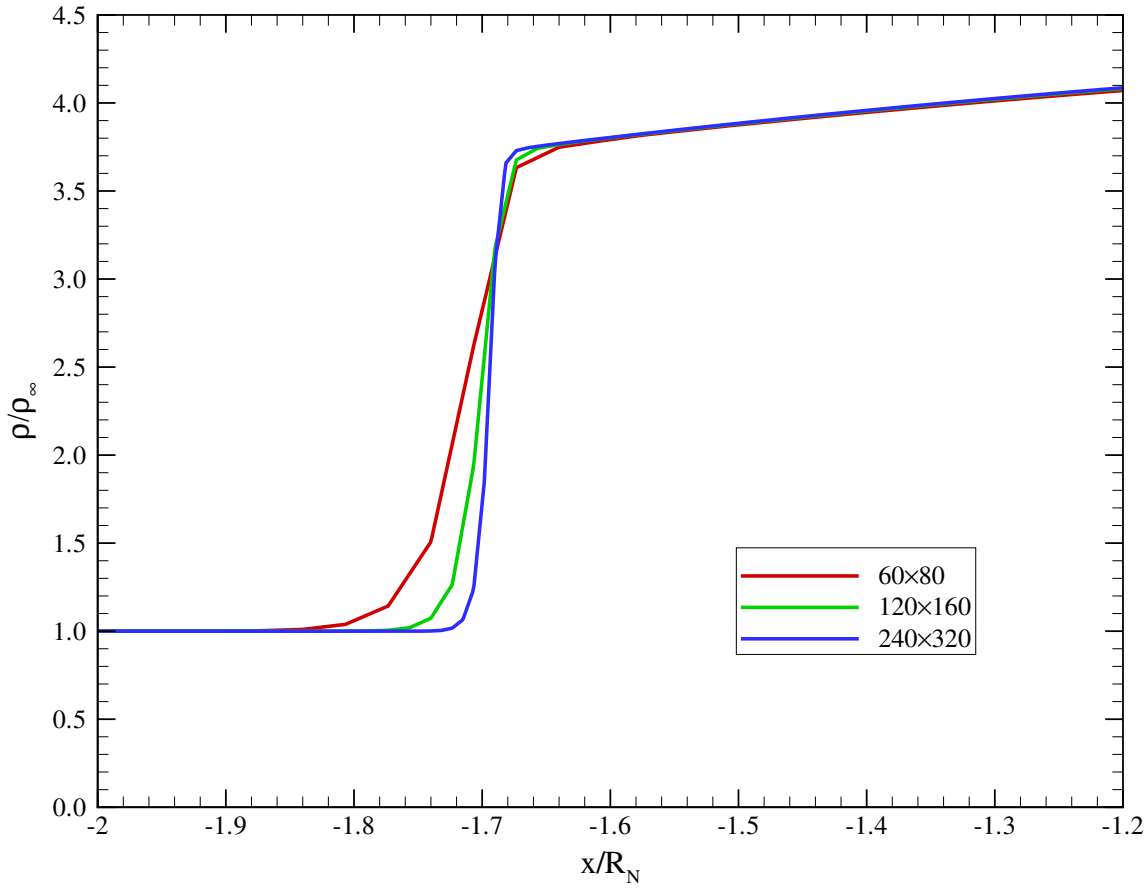


Figure 5.7: Stagnation line density profile for Mach 3 flow over a cylinder at a series of mesh resolutions.

jump which occurs across the bow shock along the stagnation line. For the series of nested meshes both the location and strength (indicated by the density jump across the shock) of the bow shock is consistent for all three simulations. Interestingly, the location of the trailing edge of the shock wave is more consistent across the series of meshes than the leading edge of the shock.

Figure 5.8 shows the shock capturing parameter, δ, along the stagnation line for the three nested meshes. Interestingly, in all cases the shock capturing parameter peaks

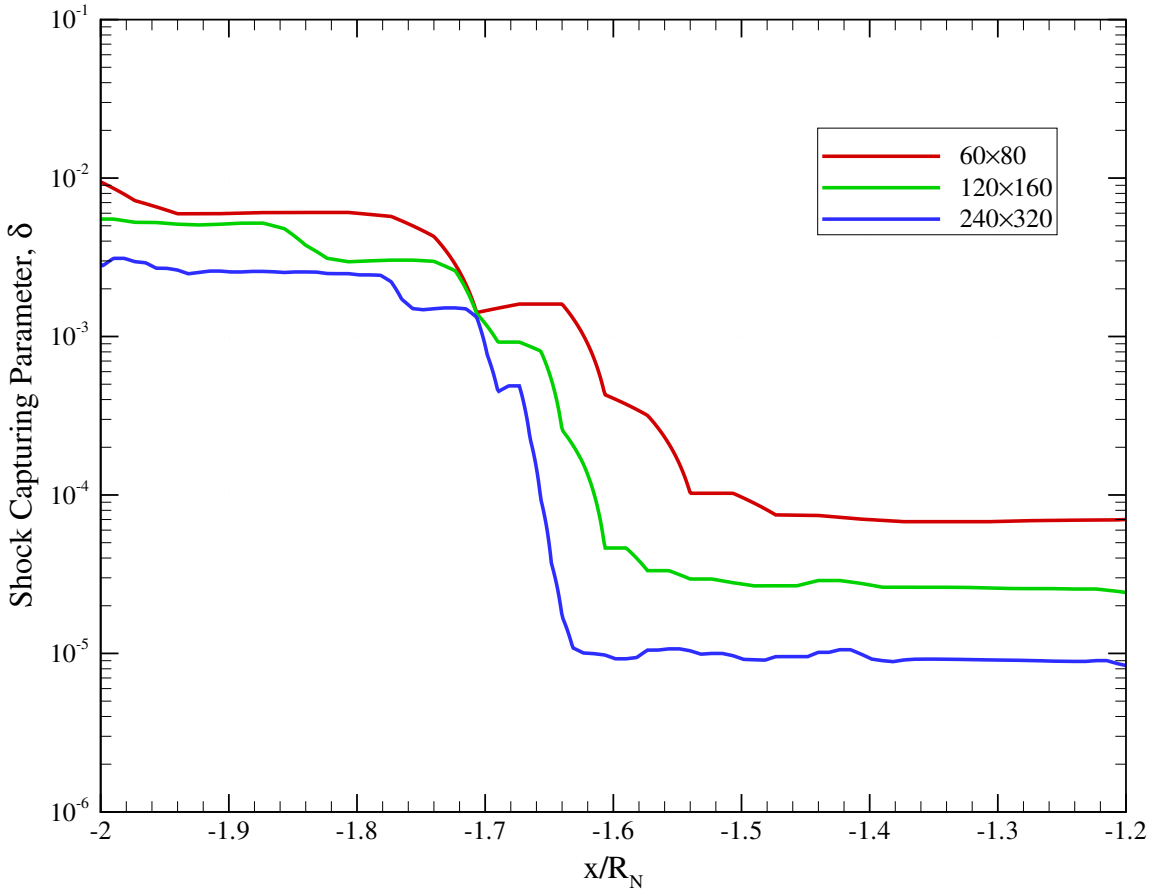


Figure 5.8: Stagnation line shock capturing parameter profile for Mach 3 flow over a cylinder at a series of mesh resolutions.

upstream of the bow shock in the uniform freestream. Upon returning to Equation (5.42) it is clear that in the uniform freestream the term behaves as $\mathcal{O}(0/0)$, and the relatively large magnitude of the parameter is a numerical artifact. Since the flow is uniform in this region the artificial diffusion term weighted by δ is inconsequential, so this behavior, albeit unsettling, does not adversely affect the quality of the solution.

The parameter decreases nearly monotonically through the bow shock and reaches a steady, low value in the post-shock stagnation region. The post-shock value of the shock capturing parameter decreases from approximately 10^{-4} to 10^{-5} with two levels of uniform mesh refinement. Since this corresponds to a factor of four reduction in the mesh spacing h, for this case δ appears to decrease superlinearly with h. For this case $\delta \propto h^{1.5}$.

5.6.1.4 Adaptive Mesh Refinement

The primary feature of this flow is the bow shock created by the cylinder. This feature is highly localized, and therefore could be captured effectively with an adaptive technique. To assess the viability of the adaptive approach for this stationary inviscid flow the simulation was repeated on the coarsest background mesh (60×80). The gradient indicator of Equation (2.1) was used to select which elements would be refined.

The simulation was initialized as before with the freestream conditions specified everywhere in the domain. The solution was then marched in time on the coarse mesh until the bow shock reached its steady location. This allowed the majority of the transient startup phase to be performed at little cost on the coarse mesh. Once the solution on this coarse mesh became steady, the adaptive mesh refinement process was initiated. The maximum level of refinement was restricted to four, which would correspond in the fine mesh regions to a uniform mesh of 480×640 elements. The resulting mesh, colored by density, is shown in Figure 5.9.

The resulting computational mesh contains 26,076 cells and 27,434 nodes for a total of 109,736 degrees of freedom. To achieve the same local mesh resolution in the vicinity of

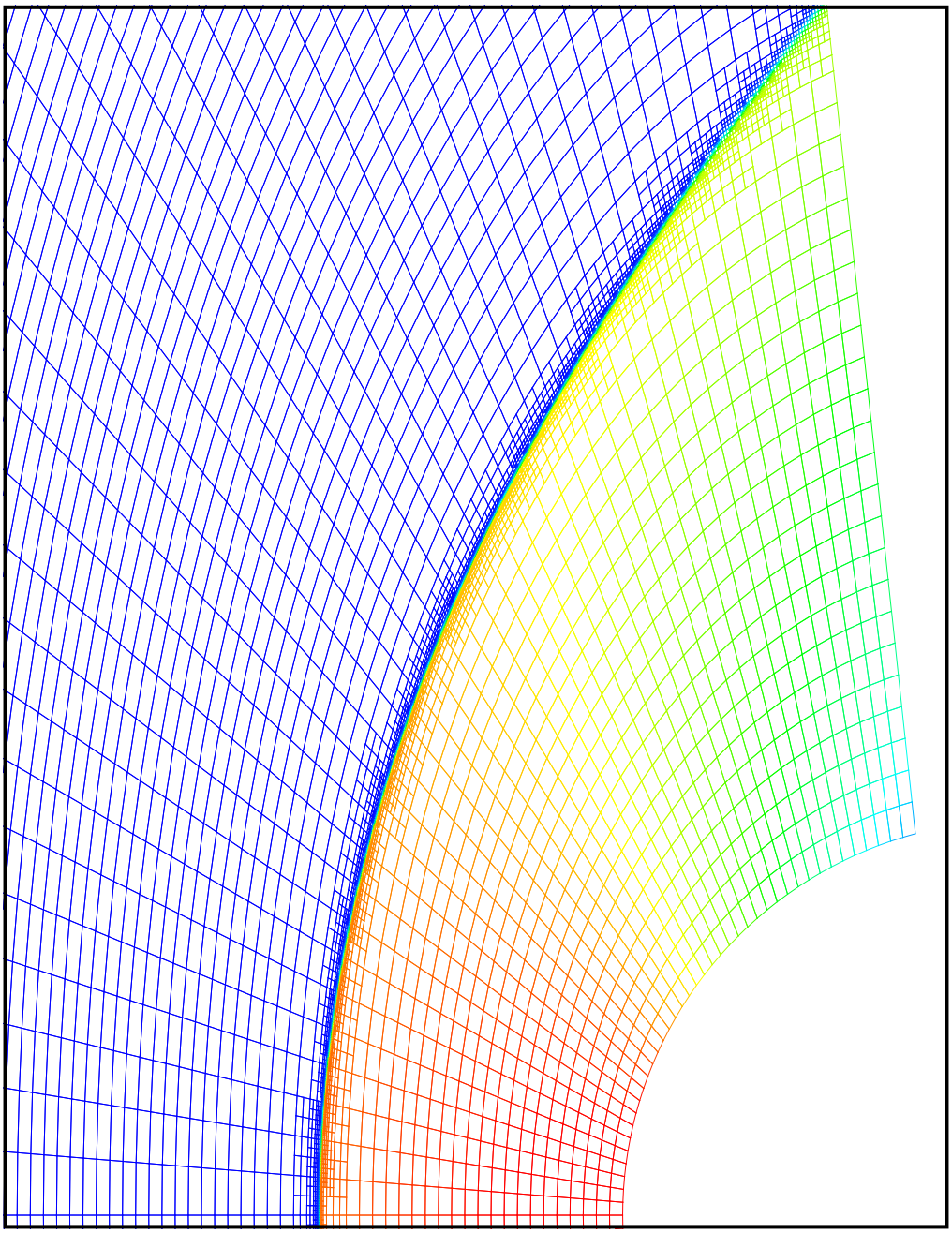


Figure 5.9: Adapted mesh capturing the bow shock for Mach 3 flow over a cylinder. The mesh is colored by the density.

the shock wave using a 480×640 mesh would require 307,200 cells (12 times as many cells as in the adapted mesh). Of course, a non-adapted mesh of this resolution would clearly afford a higher-accuracy global solution, so the comparison is not entirely fair. Still, the savings afforded by local mesh refinement are significant, especially when considering that the smaller, adapted mesh may be run quickly on a modest desktop machine.

It was mentioned previously that the shock-capturing scheme used in this work smears the shock wave across 2-3 elements when the shock is aligned with the mesh. The scheme is more diffusive when the shock and mesh are not aligned, capturing the feature smoothly across approximately four elements. The adaptive mesh refinement process does remarkably well at refining the mesh in this region to increase the resolution of the captured shock even though the background mesh is not aligned with the feature.

A common practice in finite volume simulations of hypersonic flows is to "tailor" the outer boundary of the mesh to the location of a shock wave. In this process the location of the outer boundary is moved to just upstream of the shock and its orientation is changed such that the shock is parallel to element interfaces. This necessarily introduces some distortion into the mesh which can degrade overall solution accuracy. The present adaptive procedure provides an alternate approach for accurately capturing a strong shock wave which preserves the qualities of the original mesh.

As noted in Section 5.3.2, the additional shock-capturing parameter used to regularize the problem renders the scheme locally 1^{st}-order accurate in the vicinity of shock waves. The only viable approach for increasing the solution accuracy in this case is to reduce the local mesh size substantially. The adaptive refinement procedure does exactly that. By rapidly decreasing the mesh size in the vicinity of a shock wave the overall accuracy of the solution is increased.

The stagnation line density profile presented earlier is revisited with the adaptive results in Figure 5.10. The adaptive result continues the trend that was observed previously, namely that the shock foot moves progressively downstream as the mesh is refined. The

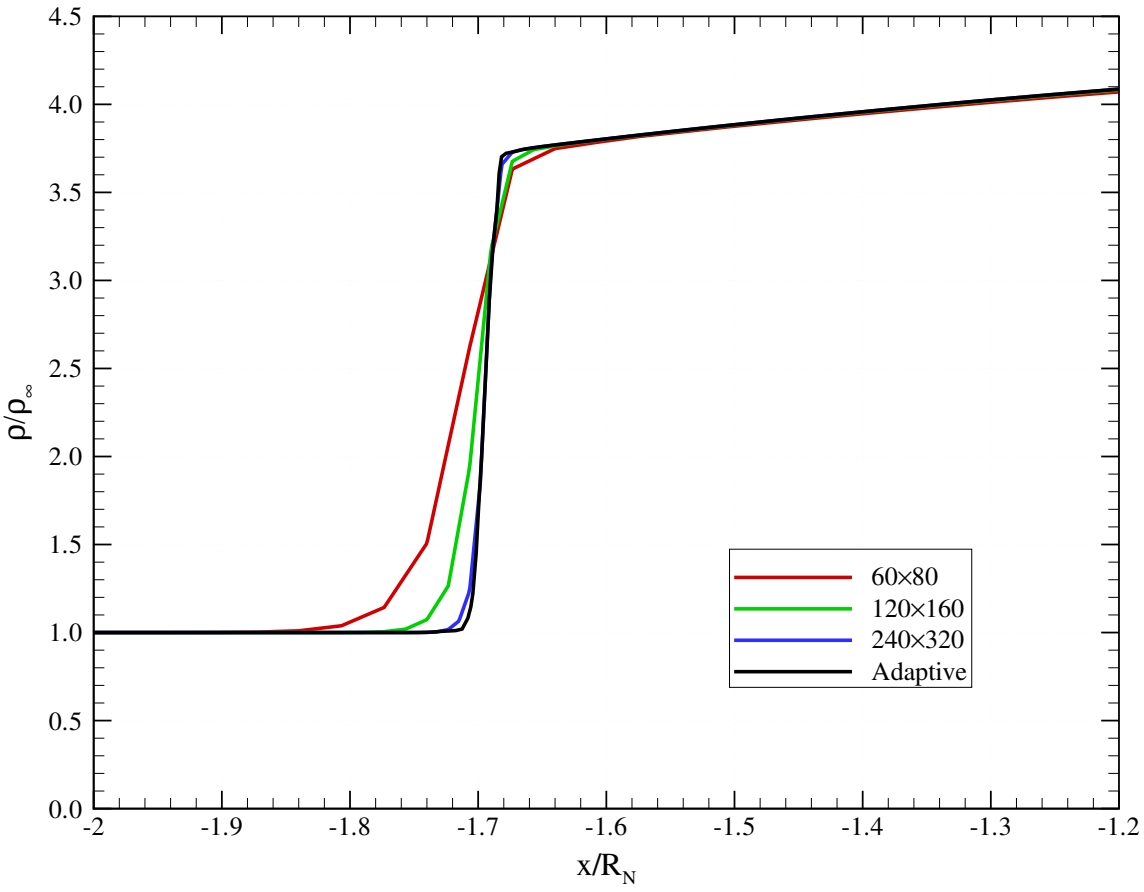


Figure 5.10: Stagnation line density profile for uniform and adapted meshes capturing the bow shock for Mach 3 flow over a cylinder.

post-shock location is essentially constant for all meshes considered. The adapted mesh, with its finer spacing in the shock region, provides the sharpest shock wave.

5.6.2 Hypersonic Flow over a Compression Ramp

This case considers laminar hypersonic flow over a 15° compression ramp. The freestream Mach number is 11.68, temperature is 64.6 K, and unit Reynolds number is 558,000 1/m. Figure 5.11 illustrates the ramp geometry and boundary conditions. The Reynolds number based on the flat plate length is Re_L =246,636 [57, 58, 59]. The

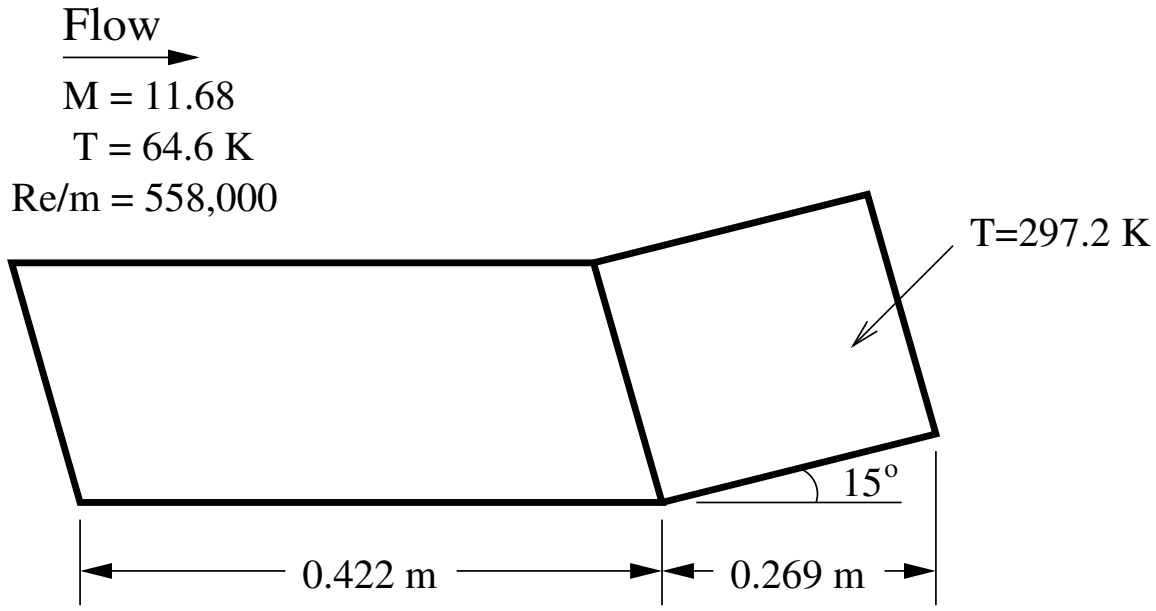


Figure 5.11: Illustration of geometry and boundary conditions for hypersonic shock ramp problem

freestream conditions for this case are explicitly listed in Table 5.3.

5.6.2.1 Motivation

Supersonic flow over a compression ramp is of interest in aerodynamic applications because it is analogous to a control surface deflecting into a supersonic flow. For this case

Table 5.3: Freestream parameters for hypersonic compression ramp [57, 59].

M_∞	Re_L	T_∞	T_w
11.68	246,636	64.6 K	297.2 K

a weak shock will develop at the leading edge of the plate due to displacement thickness effects from the viscous boundary layer. The boundary layer thickness will grow relatively quickly down the plate length due to the high edge Mach number. The compression ramp will produce an additional weak shock which is required to deflect the incoming supersonic flow. This weak shock causes a pressure increase on the compression ramp surface which can feed upstream in the subsonic portion of the boundary layer. This adverse pressure gradient, in turn, will affect the laminar boundary layer itself and can induce separation. For the case of control surface deflection the resulting pressure distribution on the compression ramp is of interest because it will dictate the performance of the control surface itself. Additionally, the heat transfer in this interaction region is also of interest because localized peaks can occur due to the laminar shock/boundary layer interaction, and these effects must be accounted for in the control surface design.

5.6.2.2 Computational Mesh

A single structured grid was used to discretize the domain and is shown in Figure 5.12. The outer boundary of the grid was created with a straight segment from the leading edge of the plate to the outflow boundary. The height of the outflow boundary was chosen such that the weak shock and subsequent Mach wave produced by the upstream portion would be fully contained within the flow domain. The left and upper boundaries are specified as freestream with essential boundary conditions. The plate itself is modeled as an isothermal no-slip wall. While not visible in the image, there is a very small region upstream of the plate leading edge which is modeled with a symmetry boundary condition. Thus, there is a slip/stick boundary condition on the velocity at the leading edge of the

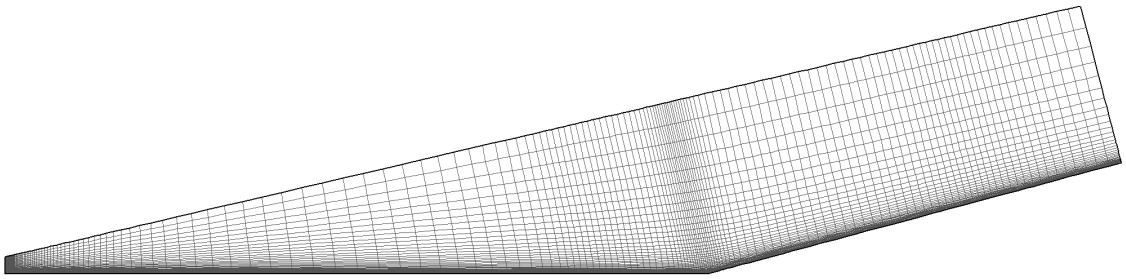


Figure 5.12: Baseline computational mesh used for hypersonic flow over a compression ramp. (Every-other point is shown)

plate. The baseline non-adapted mesh used in the simulation contains 46,680 quadrilateral elements with 47,190 nodes, yielding a discrete problem with 188,760 degrees of freedom. The mesh was partitioned into 6 subdomains and the simulation was run in parallel on a group of desktop-class machines. The partitioned mesh is shown in Figure 5.13. Note that due to the fine streamwise and normal mesh resolution used at the leading edge of the plate one of the subdomains is so physically small as to be barely visible in the Figure, although each subdomain contains roughly the same number of elements.

Figure 5.13: Partitioned mesh for hypersonic flow over a compression ramp.

5.6.2.3 Results

Figure 5.14 depicts the global flowfield for this case. The adverse pressure gradient induced by the compression ramp is evident in Figure 5.14(c). This pressure gradient

causes the boundary layer to separate upstream of the compression ramp. The recirculation region is shown in detail in Figure 5.15.

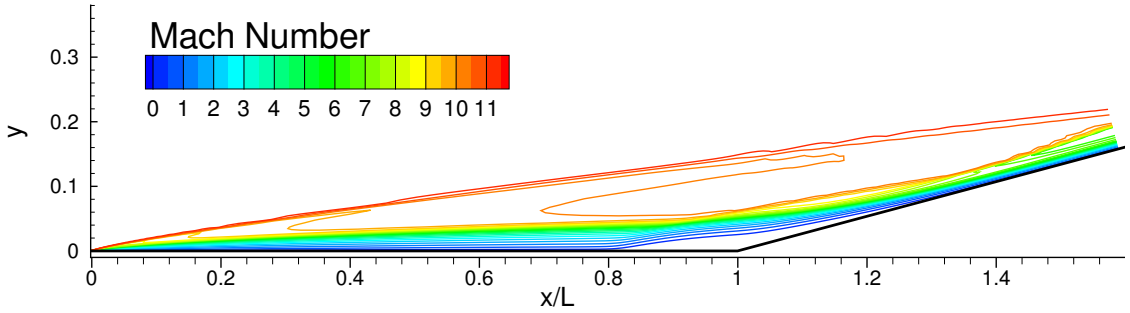
Figures 5.16 and 5.17 compares the computed skin friction coefficient and Stanton number distribution with measurements made by Holden [57]. The experimental data were obtained in the Calspan 48-inch shock tunnel. The test article was instrumented with thin-film heat transfer gages, pressure transducers, and skin friction transducers along the centerline. A range of plate widths were tested to ensure that the centerline data were not adversely affected by three-dimensional expansion effects [57].

The surface shear is an excellent indicator of the onset of separation which occurs upstream of the compression ramp corner (see Figure 5.15). At the separation point the surface shear vanishes. The attached upstream flow produces a positive shear while the flow in the recirculation region produces a negative shear. Figure 5.16 plots the nondimensional skin friction coefficient versus the nondimensional distance from the leading edge of the plate. The skin friction coefficient is defined as

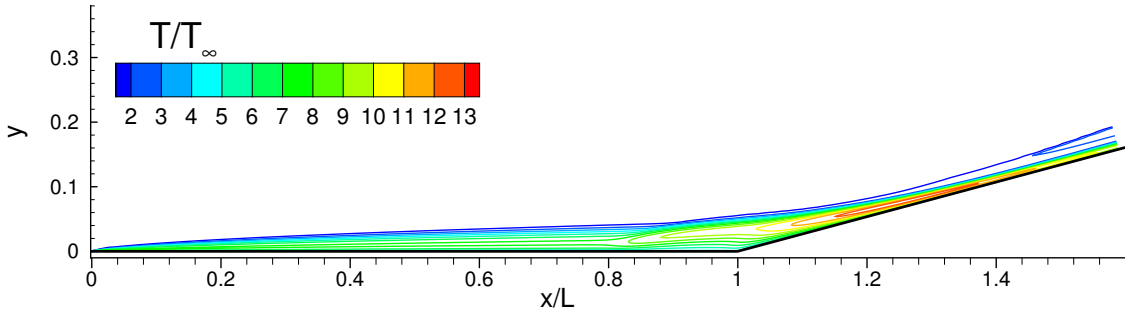
$$C_f = \frac{\tau_w}{\frac{1}{2}\rho_\infty U_\infty^2} \tag{5.70}$$

where τ_w is the shear stress which is nondimensionalized with the freestream dynamic pressure. The experimental and computed values are in general agreement, and the magnitude of the shear is in excellent agreement in the recirculation region (and hence the strength of the recirculation). Similar results were reported by Lillard and Dries with a completely different flow solver [60].

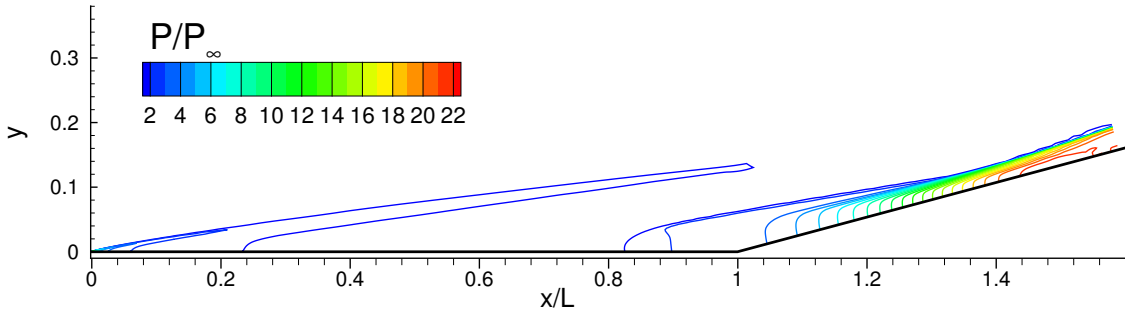
The surface heat transfer is critically important because of the severe heating which can occur in the reattachment region on the compression ramp. In this region the edge of the boundary layer is subject to a compression fan which markedly thins the boundary layer. The resulting surface heat transfer obtains a local maximum. As previously discussed, the compression ramp serves as a conceptual model for a control surface deflected into a hypersonic stream. In this application it is critically important to understand the magnitude of the reattachment heating because it provides the design environment for the



(a) Mach Contours



(b) Temperature Contours



(c) Pressure Contours

Figure 5.14: Illustration of flowfield for hypersonic shock ramp problem

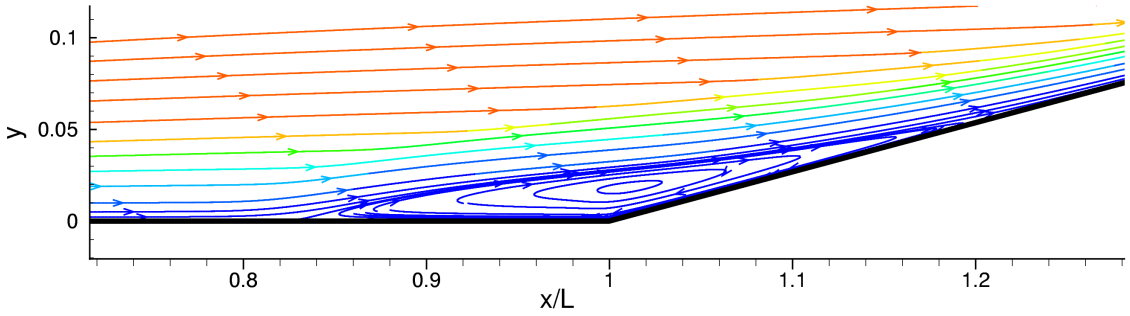


Figure 5.15: Compression ramp recirculation region

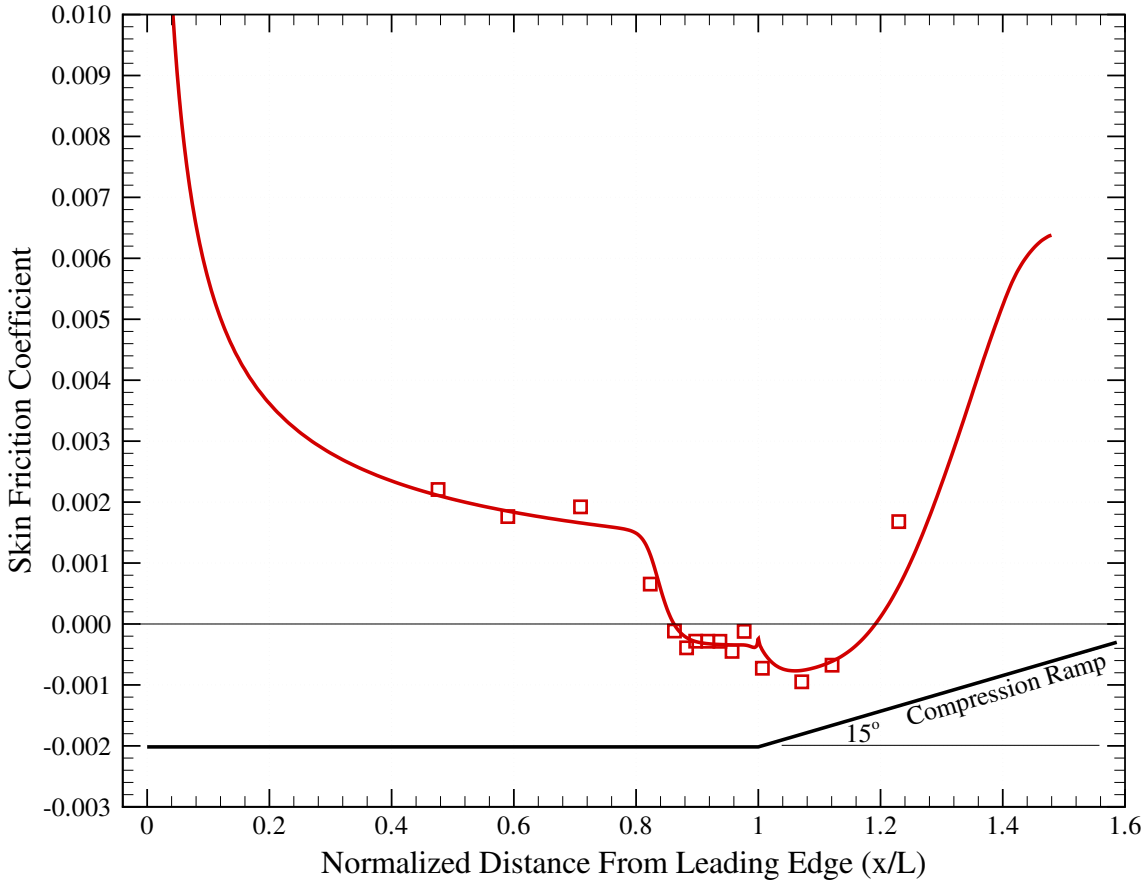


Figure 5.16: Skin friction coefficient (solid line) comparison with experimental data (point values) for hypersonic shock ramp problem

thermal protection system on the control surface. An example of this is the "body flap" on the Space Shuttle Orbiter. On the body flap the thermal protection system silica tiles are approximately four times thicker than those immediately upstream because of the increased reattachment heating which occurs when the control surface is deflected.

In Figure 5.17 the computed and measured heat transfer are compared. The wall heat transfer, q_{wall}, is nondimensionalized by means of the Stanton number

$$St = \frac{q_{wall}}{\rho_\infty U_\infty c_p \left(rT_0 - T_w\right)} \tag{5.71}$$

where T_0 is the freestream total temperature, T_w is the surface temperature of the model, ρ_∞ and U_∞ are the freestream density and velocity, and c_p is the freestream specific heat at constant pressure. The recovery factor, r, is assumed equal to one in this case.

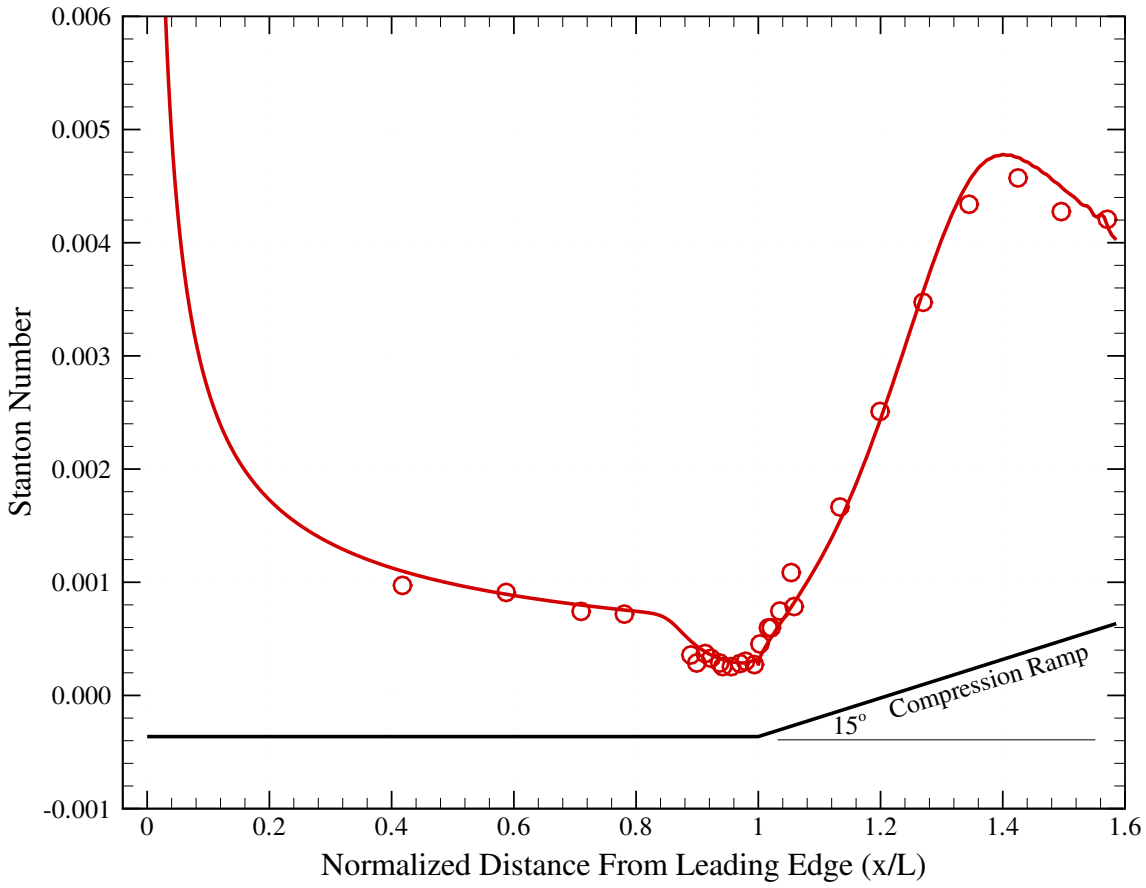


Figure 5.17: Stanton number comparison with experimental data for hypersonic shock ramp problem

5.6.2.4 Convergence

The simulation was initialized from freestream values and advanced in time using the geometric time advancement scheme described in equations (5.68) and (5.69). An initial time step of 5×10^{-6} was used and a geometric growth factor of $r = 1.1$ was employed. The maximum time step was limited to $\Delta t_{\text{max}} = 0.5$.

The normalized unsteady residual versus time step is shown in Figure 5.18. After

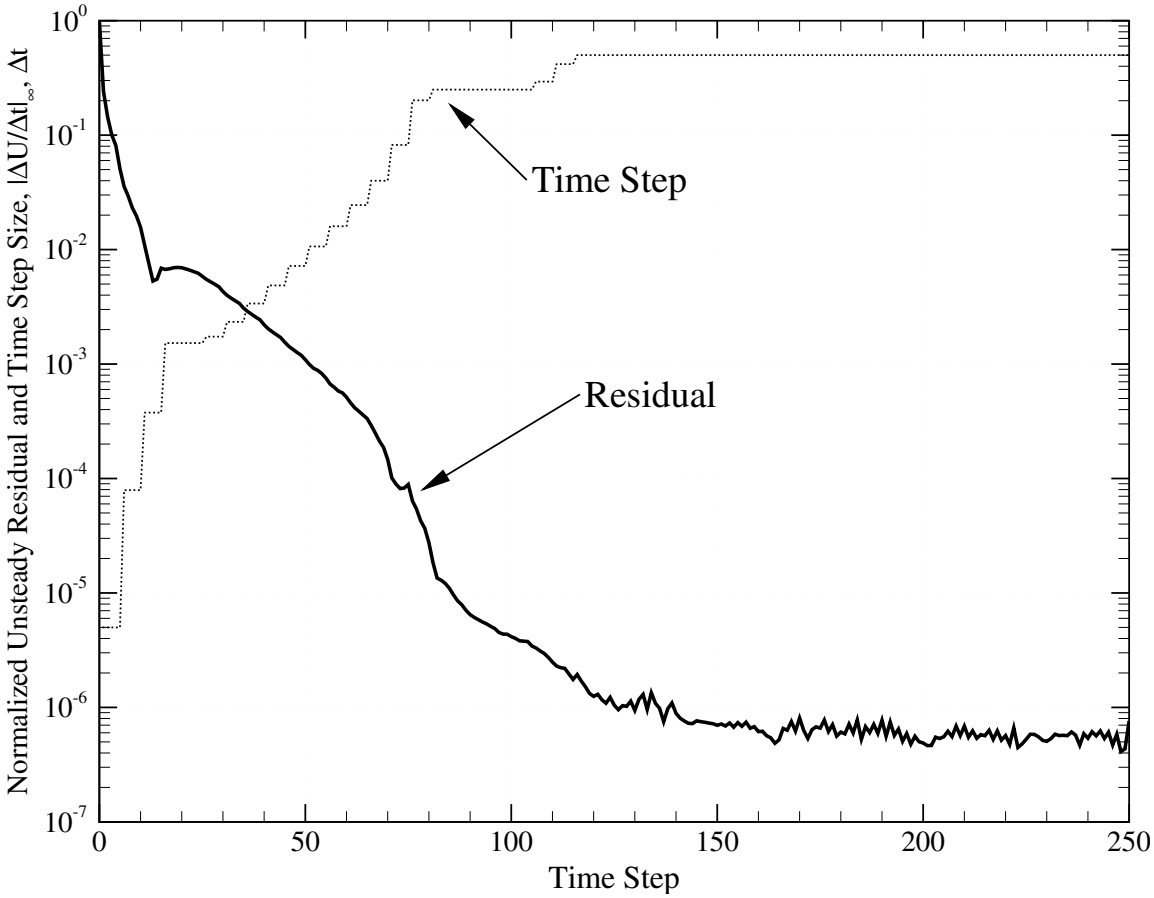


Figure 5.18: Time step convergence history for hypersonic flow over a compression ramp.

an abrupt initial decrease the residual is seen to stagnate at a value of approximately 10^{-6}. This is in contrast to the results seen previously for the case of inviscid flow over a cylinder.

It should be noted that during this residual stagnation period the solution is visibly steady and both the surface pressure and heat transfer distributions are converged. Therefore, to investigate the source of the residual stagnation the solution at two different time steps, 200 and 220, was differenced at each node in the mesh and examined. This procedure indicated that changes on the order of 5–10% occur at the leading edge of the plate in the vicinity of the slip/stick boundary condition singularity.

The nondimensional time step used in the simulation is also shown in the figure. Consistent with Equation (5.68), the time step is seen to increase geometrically with decreasing unsteady residual. Initialized from freestream values, the scheme converges to its steady-state in approximately 150 time steps. During the first half of this transient the inviscid flow sets up and produces high pressure on the surface of the wedge. As discussed previously, this induces boundary layer separation, which initiates in the corner and feeds upstream. During the second half of the transient the separation size gradually increases as the separation point continues to move upstream and the reattachment point moves downstream until steady-state is reached.

5.6.2.5 Adaptive Mesh Refinement

It is clear from the previous global flowfield images that the primary features of this flowfield are the separated recirculation region and the weak shock which develops from the compression ramp surface. These features are highly localized. The structured grid used in the previous simulations necessarily introduces additional resolution in benign regions of the flow such as downstream of the leading edge shock and upstream of the compression ramp shock.

The primary feature of this flow is the viscous/inviscid interaction set up by the separated region. The inviscid portion of the domain is adiabatic, but there are considerable non-adiabatic effects in the boundary layer. In the adiabatic region the total enthalpy in the flow, H, is constant, while H will vary appreciably in the viscous/inviscid interaction region. Accordingly, a refinement feature indicator was devised by constructing $\|\boldsymbol{\nabla} H\|$,

the magnitude of total enthalpy gradient on each element in the domain.

Figure 5.19 details the static temperature and resulting adapted mesh in the vicinity of the compression corner for this case. The initial mesh corresponds to a twice-coarsened

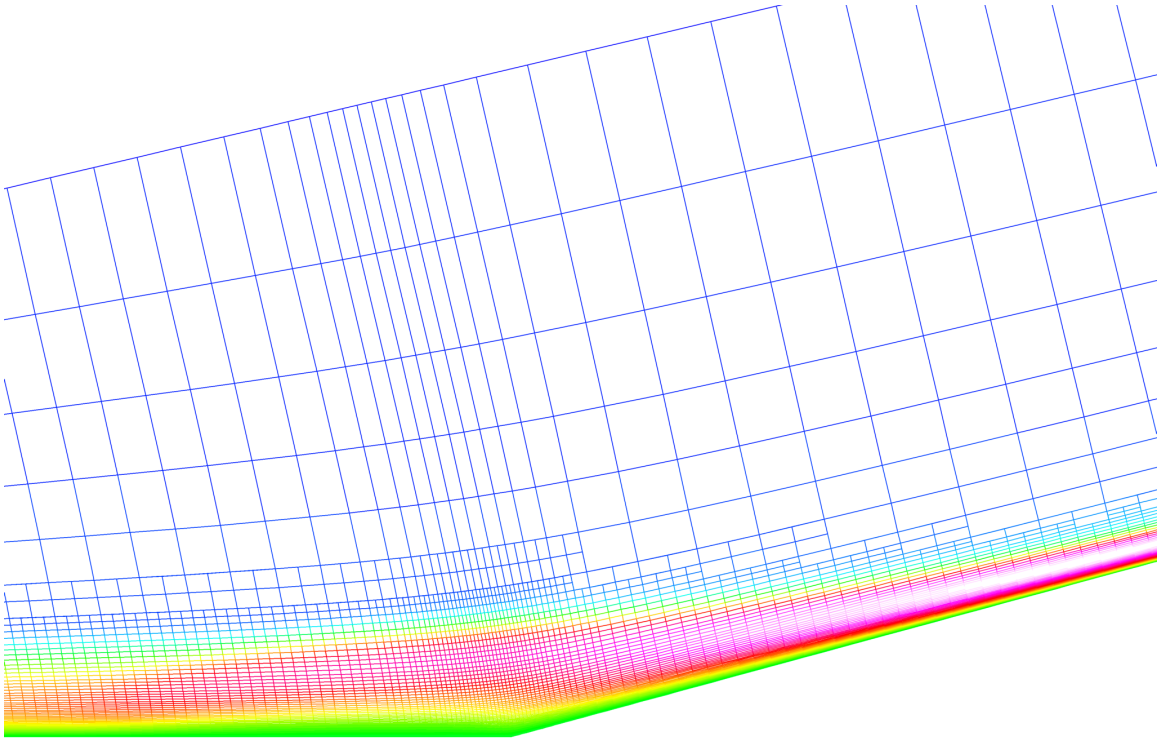


Figure 5.19: Adapted mesh and static temperature contours for hypersonic flow over a compression ramp.

version of the baseline mesh used in the previous studies. This coarse mesh contains only 2,940 elements and 3,069 nodes. At each stage in the adaptive refinement algorithm the feature indicator is computed for each active element. The mean (f_{m}) and standard deviation (σ) are then computed from each element contribution. Elements which contain values greater than $\left(f_{\mathrm{m}} + \frac{\sigma}{2}\right)$ are selected for refinement, while those with less than $(f_{\mathrm{m}} - 2\sigma)$ are coarsened.

The final adapted mesh contains approximately 30,000 nodes, which is a 35% reduction from the baseline mesh. The flowfield shown in Figure 5.19 is qualitatively similar

to the uniform mesh result. A quantitative comparison is shown in Figure 5.20 in which the Stanton number from the adaptive and uniform simulations are compared. The two results agree extremely well, with only a small discrepancy in the peak heating value which occurs downstream of the reattachment point.

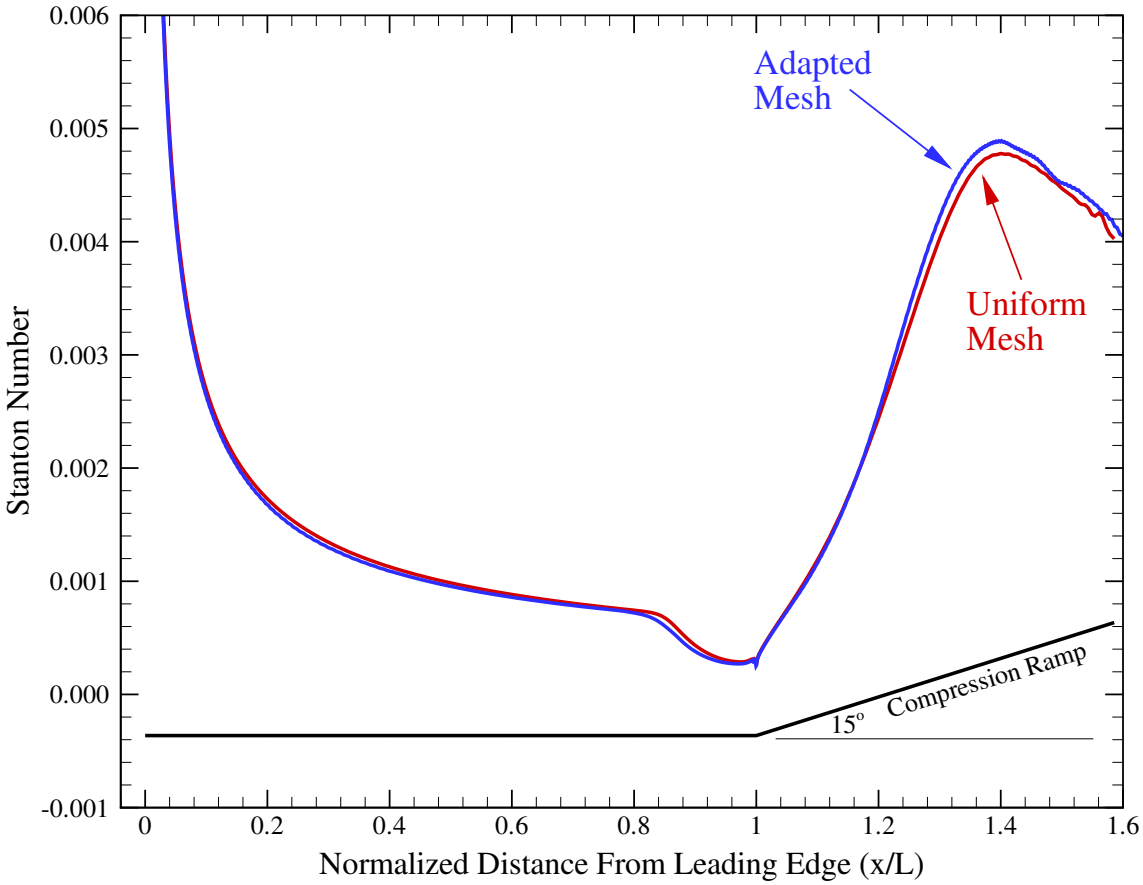


Figure 5.20: Stanton number for uniform and adapted meshes for hypersonic flow over a compression ramp.

5.6.3 Hypersonic Flow over an Axisymmetric Hollow Cylinder-Flare

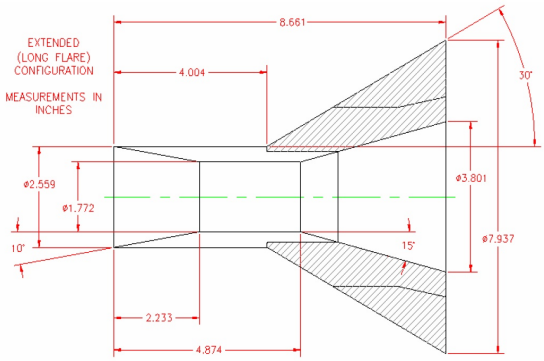
The 15° compression ramp studied in the previous section produces a strong viscous-inviscid interaction in which the inviscid pressure rise caused by the compression ramp induces boundary layer separation and a corresponding recirculation region upstream of the compression surface. This recirculation region produces a separation shock which does not interact significantly with the model surface. In this section a stronger interaction on a hollow cylinder-flare model is considered. The availability of high-quality experimental data makes this case valuable for validating the present finite element model.

5.6.3.1 Background

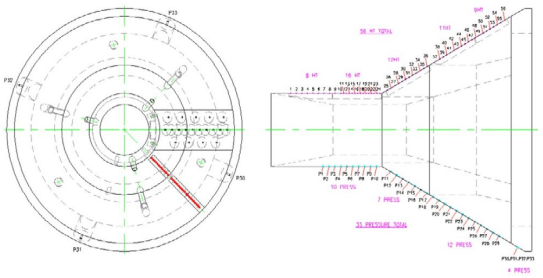
The configuration examined is an axisymmetric hollow cylinder with a flare inclined at 30°. (The use of axisymmetric models removes the question of width and edge effects which are an issue for two-dimensional configurations and are particularly problematic for shock interaction problems.) The resulting shock/shock and shock/boundary layer interaction produces a large, localized peak in heat transfer on the model surface. Experimental data were obtained for this configuration at the Calspan-University of Buffalo Research Center (CUBRC) Large Energy National Shock (LENS) Facility. An image of the model and schematics depicting the dimensions and instrumentation layout are shown in Figure 5.21. The freestream conditions for this case are listed in Table 5.4.

The model was instrumented with a series of thin-film heat transfer gages and piezoelectric pressure transducers. The reported accuracy of the heat transfer measurements is $\pm 5\%$. The model was tested in pure Nitrogen to minimize chemical nonequilibrium effects in the flow, which would be appreciable in air at the high freestream enthalpies used in the test [61]. The collected data are extremely valuable for validation of numerical schemes because the flow conditions are such that laminar flow results with minimal chemistry effects. Hypersonic ground test data in which transition and turbulence or chemical nonequilibrium effects are important are difficult to compare with numerical simulation because the freestream conditions and boundary layer state are often difficult to characterize.

(a) Test Article.



(b) Schematic. All dimensions are inches.



(c) Instrumentation Layout.

Figure 5.21: Hollow cylinder test article and dimensions.[61]

Table 5.4: Freestream parameters for hypersonic hollow cylinder-flare benchmark [62].

M_∞	Re_L	T_∞	T_w
10.3	25,347	120.4 K	295.2 K

Previous numerical studies by Gnoffo [62] and MacLean [63] (both of whom assessed the influence of thermal nonequilibrium) have indicated that the assumption of a calorically perfect gas is valid for this case. The transport properties for the flow are given via Sutherland's law with the assumption of constant Prandtl number as described in (5.18). For perfect Nitrogen, Sutherland's law takes the form

$$\mu_{N_2} = 1.399 \times 10^{-6} \frac{T^{\frac{3}{2}}}{T + 106.67} \ \mathrm{Pa \cdot s} \tag{5.72}$$

5.6.3.2 Computational Mesh

The computational mesh used for this case is shown in Figure 5.22. It uses two structured grid blocks to encompass the external flow and a portion of the interior just below the sharp leading edge. Including the interior portion of the domain eliminates the slip/stick velocity boundary condition singularity present in the case of the compression ramp. The outer boundary of the grid was tailored such that the wall-normal spacing in the reattachment area is minimized, thus allowing for focused resolution in this high gradient region. As in the previous flat plate case, the height of the outflow boundary was chosen such that the oblique shocks produced by the cylinder displacement thickness and flare would be fully contained within the flow domain. The left and upper boundaries are specified as freestream with essential boundary conditions, and the cylinder-flare is modeled as an isothermal no-slip wall. The baseline non-adapted mesh used in the simulation contains 45,906 quadrilateral elements with 46,600 nodes, yielding a discrete problem with 186,400 degrees of freedom.

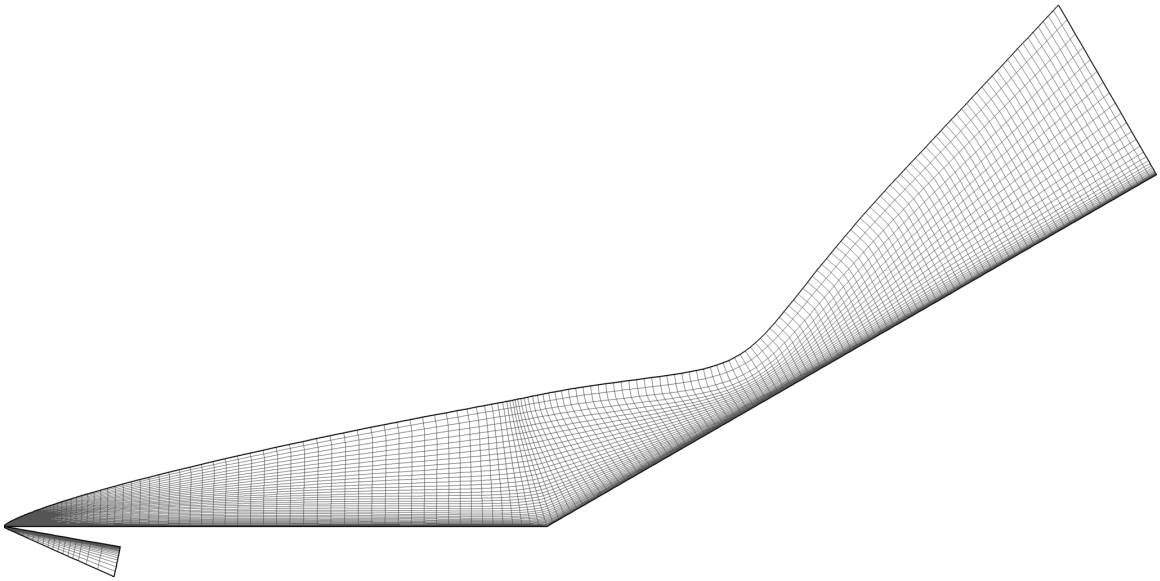


Figure 5.22: Baseline computational mesh used for hypersonic flow over a hollow cylinder-flare. (Every-other point is shown)

5.6.3.3 Results

Figures 5.23–5.25 depict the global flowfield for this case. All figures clearly depict a weak leading edge shock caused by boundary layer displacement effects which merges with a separation shock. This merged shock then impinges on the flare surface. These features will be discussed in more detail in the context of the specific field variables shown in the figures.

Several important flow features are evident from the static temperature field shown in Figure 5.23. A weak shock develops at the leading edge of the hollow cylinder due

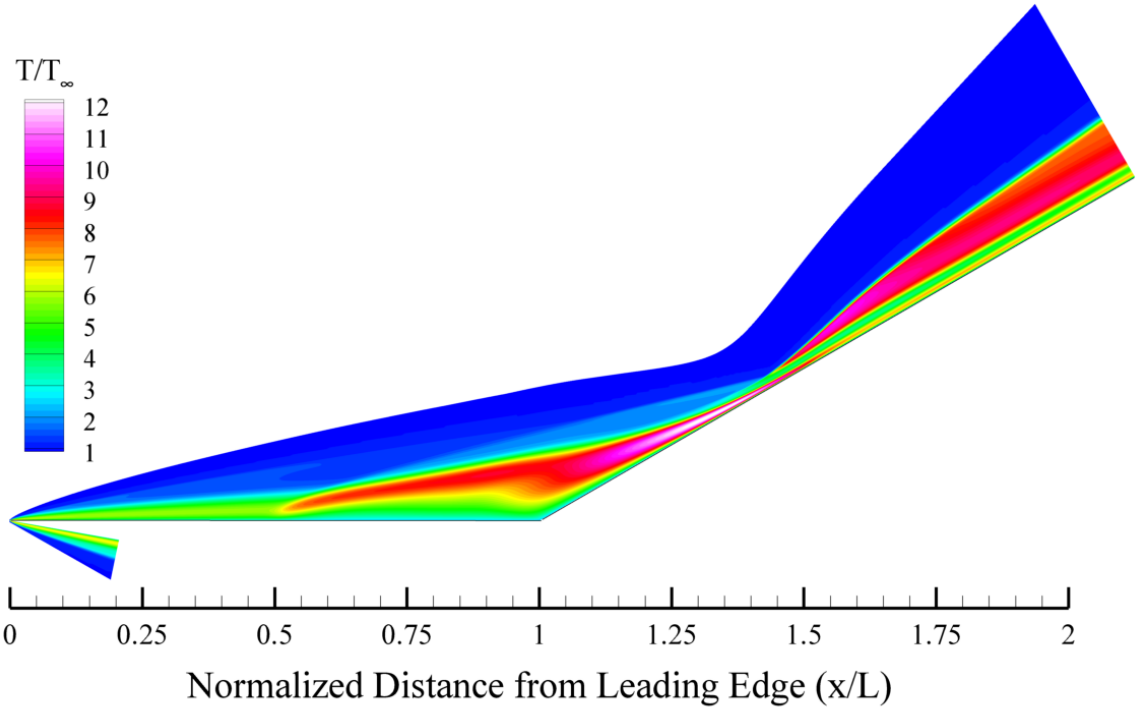


Figure 5.23: Illustration of flowfield for hypersonic hollow cylinder-flare shock interaction problem: nondimensional static temperature.

to boundary layer displacement effects. The flow separates at approximately (x/L) of 0.5 and creates a separation shock which coalesces with the leading edge shock. The separation/leading edge merged shock is then seen to impinge on the conical section in the reattachment region. The strong temperature gradient in the reattachment region at (x/L) of 1.4 is clearly evident. The peak temperature in the reattachment region is 1450 K (approximately 12 times the freestream value). Clearly these elevated temperatures would cause significant excitation and dissociation of O_2 in air, however these effects are mitigated by using N_2 as the test gas.

Figure 5.24 shows the flowfield nondimensional static pressure for this case. There is clearly a strong, local increase in pressure in the interaction/reattachment region. Further,

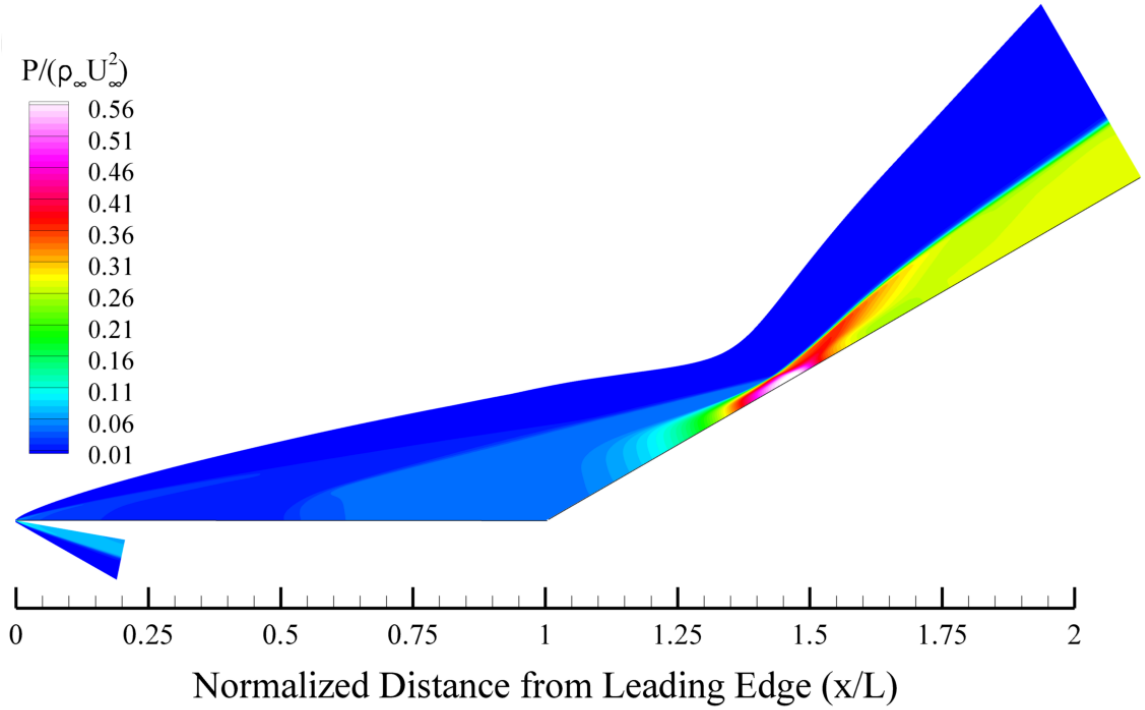


Figure 5.24: Illustration of flowfield for hypersonic hollow cylinder-flare shock interaction problem: nondimensional static pressure.

the adverse pressure gradient on the forward portion of the cone is evident. It is this adverse pressure gradient that induces boundary layer separation. In turn, the separated boundary layer induces a shock wave which interacts with the reattachment region, resulting in a tight coupling between the size of the separation region, strength of the separation shock, and the magnitude of the adverse pressure gradient. In this way this problem serves as an excellent test for a numeric scheme because an error in modeling any one of these features can be magnified by the inherit nonlinear couplings in the flowfield response.

Figure 5.25 shows the Mach number distribution. The flowfield is largely supersonic with the exception of the recirculation region. The weak leading edge shock decel-

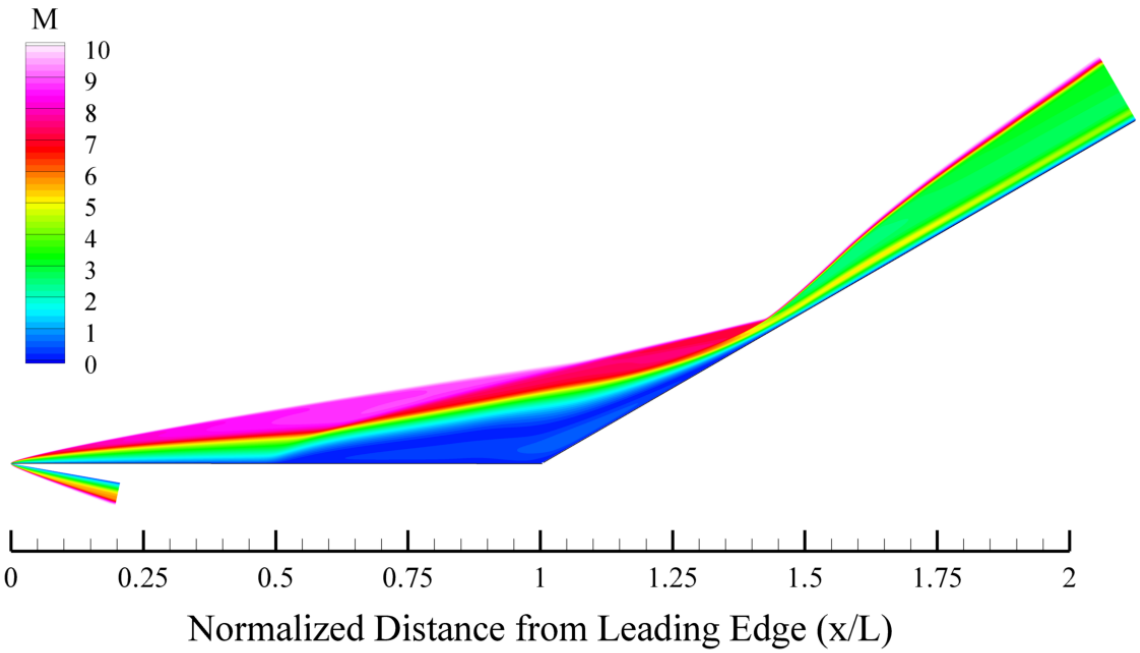


Figure 5.25: Illustration of flowfield for hypersonic hollow cylinder-flare shock interaction problem: Mach number.

erates the flow from Mach 10.3 to Mach 9, and the conical section shock decelerates the flow to approximately Mach 4. The outflow is supersonic in all but the extreme near-wall region, hence the use of an extrapolation outflow boundary condition is justified.

Figure 5.26 shows a comparison between measured and computed heat transfer and pressure coefficients. The heat transfer coefficient is defined as

$$C_H = \frac{q_{wall}}{\frac{1}{2}\rho_\infty U_\infty^3} \tag{5.73}$$

which nondimensionalizes the wall heat flux by the freestream maximum energy flux. The pressure coefficient is defined as

$$C_p = \frac{P - P_\infty}{\frac{1}{2}\rho_\infty U_\infty^2} \tag{5.74}$$

which nondimensionalizes the difference between the local and freestream pressure by the freestream dynamic pressure.

Both the trends and magnitudes of the experimental data are well captured by the numerical solution. Separation is clearly evident at (x/L) of 0.5 as indicated by the increase in pressure coefficient and corresponding decrease in surface heat transfer. The extreme wall-normal temperature gradients in the reattachment region is reflected in increased heat transfer to the surface. The measured peak heat transfer exceeds the computed value. However, similar behavior was seen by MacLean [63] with two completely different flow solvers. This lends credibility to the current numerical results. During the transient startup phase, prior to achieving steady-state, the forward extent of the separation region increases while the location of the reattachment moves downstream. The peak heating in the reattachment region decreases as the extent of the separated region approaches steady-state. This transient behavior is obviously present in the experimental configuration as well. Additionally, obtaining a steady response for data sampling is difficult. This is especially the case in the reattachment region, so it is possible that the experimental data correspond to a snapshot of a slowly evolving transient flowfield.

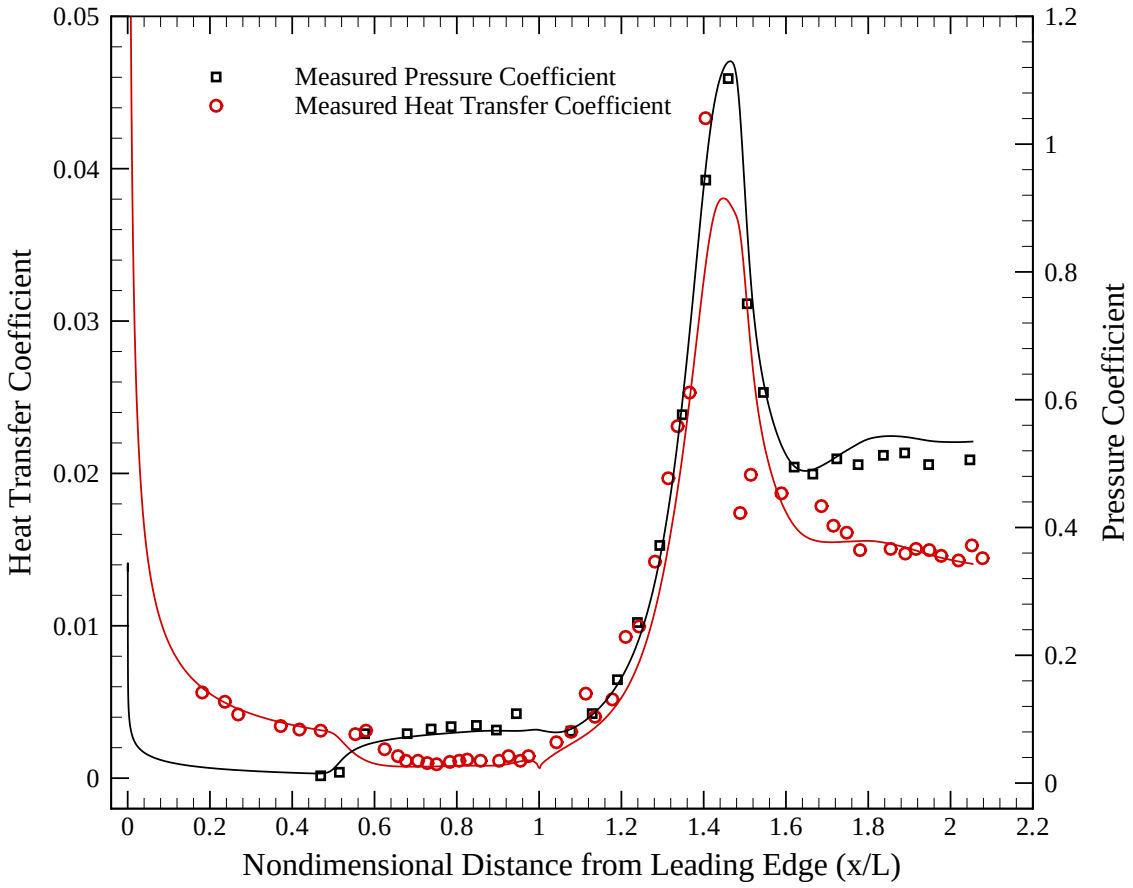


Figure 5.26: Comparison of measured and computed heat transfer and pressure coefficients for hollow cylinder-flare configuration.

5.6.3.4 Adaptive Mesh Refinement

The adaptive procedure outlined previously for the case of hypersonic flow over a compression ramp is again applied to this case. The total enthalpy gradient is used as a feature indicator to focus mesh refinement in the viscous/inviscid and shock/boundary layer interaction regions. The adapted mesh and static temperature field are shown in Figure 5.27. The background mesh is visible in the far-field portion of the domain and corresponds to a

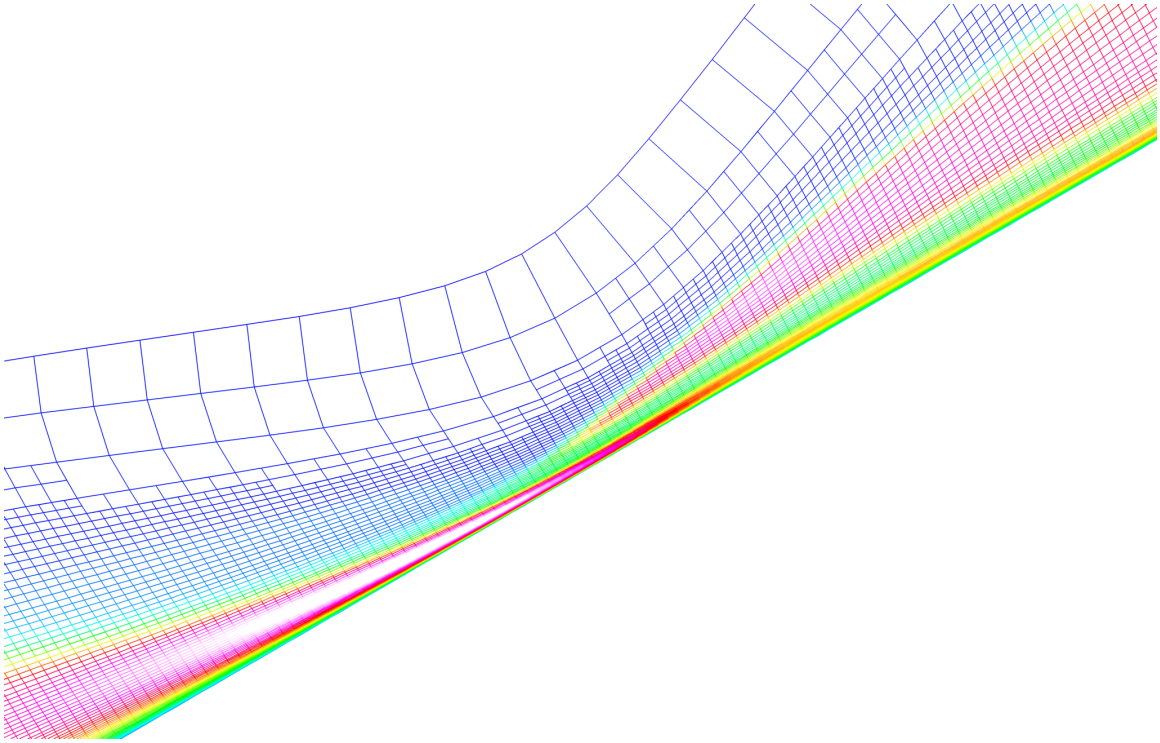


Figure 5.27: Adapted mesh and static temperature contours for the reattachment region of a hypersonic flow over a hollow cylinder-flare.

twice-coarsened version (containing a factor of 16 fewer elements) of the baseline, uniform mesh. The adapted mesh contains approximately 30% fewer degrees of freedom than the uniform mesh. The surface pressure and heat transfer for the adapted mesh are visually identical to the uniform case and thus are not shown here.

5.6.4 Hypersonic Flow over an Axisymmetric Double Cone

Another experimental case with extensive data is that of a geometrically axisymmetric biconic, or double cone model. The double cone has been extensively studied and computationally modeled because of the complex shock interaction structure that results from the compound geometric angle.

Previous research has shown that this simple geometry can yield a very complex and potentially unsteady flowfield response [64, 65]. Initial experimental testing was performed in a Nitrogen stream at CUBRC for a range of freestream Reynolds numbers [66]. Subsequent analysis by Nompelis et al. [64] showed that certain aspects of the experimental results could be best explained by accounting for vibrational nonequilibrium in the freestream. This observation was a result of detailed analysis which accounted for the nonequilibrium expansion within the nozzle to arrive at the freestream conditions which were then fixed as inflow conditions for analysis of the double cone. These observations are in agreement with previous experimental and computational investigations of double cone flow performed by Olejniczak [67] which highlighted the sensitivity of these flows to chemical nonequilibrium effects.

Following this discovery, subsequent testing was performed at the Arnold Engineering Development Center (AEDC) Hypervelocity Wind Tunnel Number 9 (HVWT9), which is located in White Oaks, Maryland. This facility provides relatively long run times (on the order of seconds) which allows for data to be obtained for a long period of time at a range of flow conditions. The tunnel also uses Nitrogen as its working fluid, hence the results obtained at CUBRC were of interest to AEDC test engineers. The same model was tested in HVWT9 to investigate the potential impacts of vibrational nonequilibrium. Contrary to the CUBRC experience, however, flow in the test section of HVWT9 was found to behave as a calorically perfect gas for all tested conditions. Also, significant unsteadiness was observed in the flowfield [65]. It is hypothesized here that the perfect-gas nature of the flow in the AEDC facility is due to intrinsic differences in the two facilities. (LENS

is a shock-driven impulse facility while HVWT9 is a blow-down tunnel.) The source of unsteadiness will be considered further in this work and is the subject of Section 5.6.4.2.3.

5.6.4.1 CUBRC Blunt Double Cone

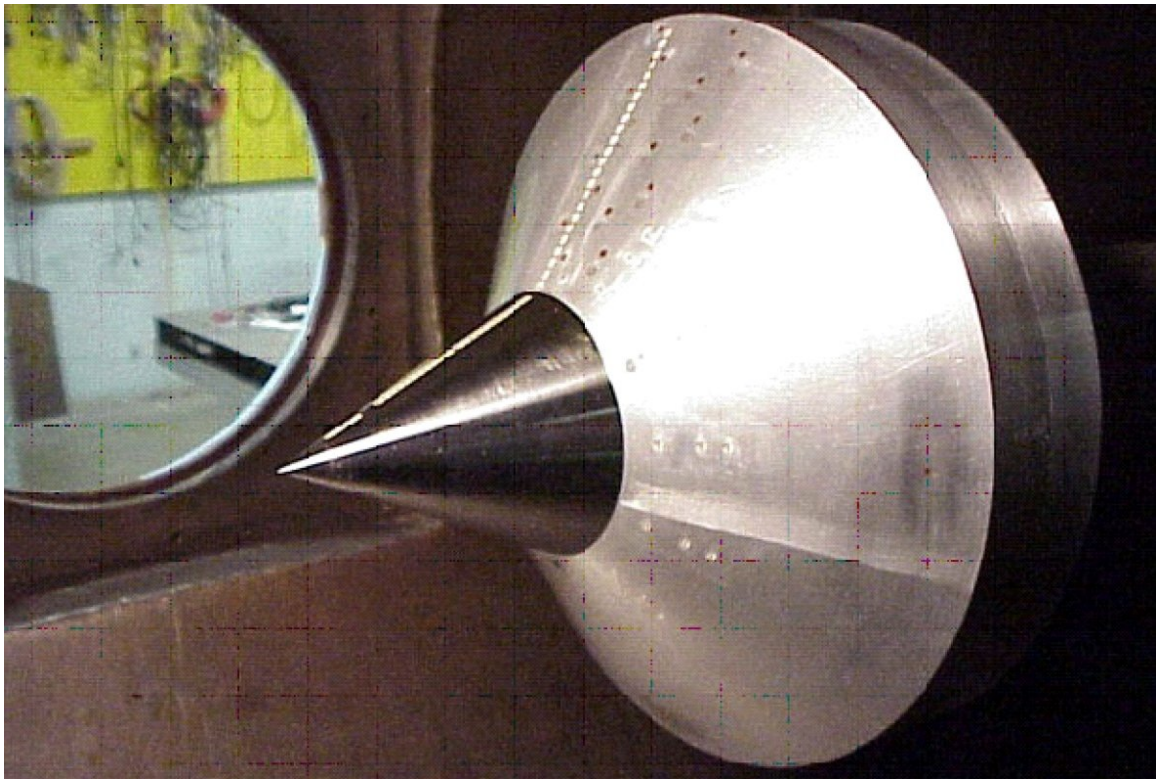
For the CUBRC code validation database, the double cone was selected to have a 25° and 55° primary and secondary half-angle, respectively. This particular choice is interesting in that the forecone 25° half-angle admits an attached oblique shock, while the 55° aftcone half-angle is sufficiently steep that an attached shock is not possible. Therefore, the aftcone produces a detached bow shock which will be intersected by the forecone oblique shock.

Code validation studies for four separate nose configurations are available. Here, the blunt nose tip radius of 6.35 mm (0.25") was selected to provide validation data for the finite element scheme at the freestream conditions specified in Table 5.5. It has previously been shown that the test data are marginally influenced by the effect of vibrational nonequilibrium for this case [63, 64]. This will be discussed further in the presentation of results. The resulting steady-state flowfield for this case is depicted in Figure 5.29.

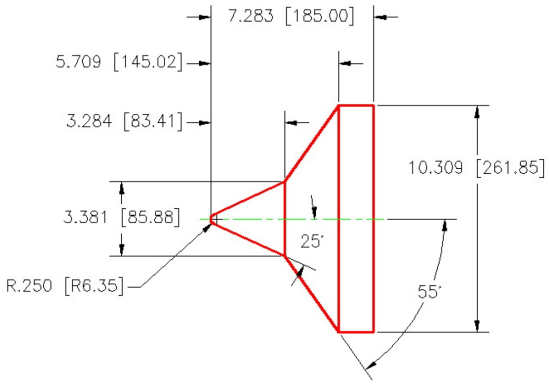
Table 5.5: Freestream parameters for hypersonic blunt double cone benchmark [63].

M_∞	Re_D	T_∞	T_w
12.43	53,666	107 K	297 K

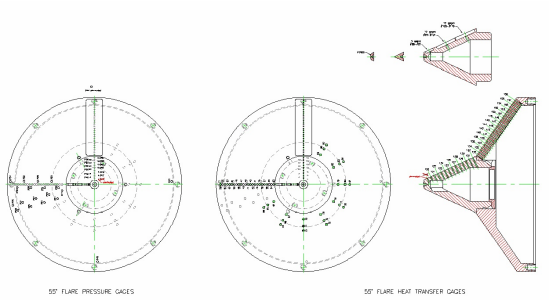
Figure 5.29(d) shows a computed schlieren image for the aforementioned case and depicts details of the complex shock interaction structure. The image was generated by plotting the magnitude of the density gradient with a gray-scale color map. As in schlieren photography, strong shock waves create a relatively larger density gradient and appear darker in the image. This is particularly the case in the interaction region, where a nearly normal is shock set up by the 55° cone. The strength of this shock decreases away from the interaction region as it becomes increasingly oblique.



(a) Test Article.

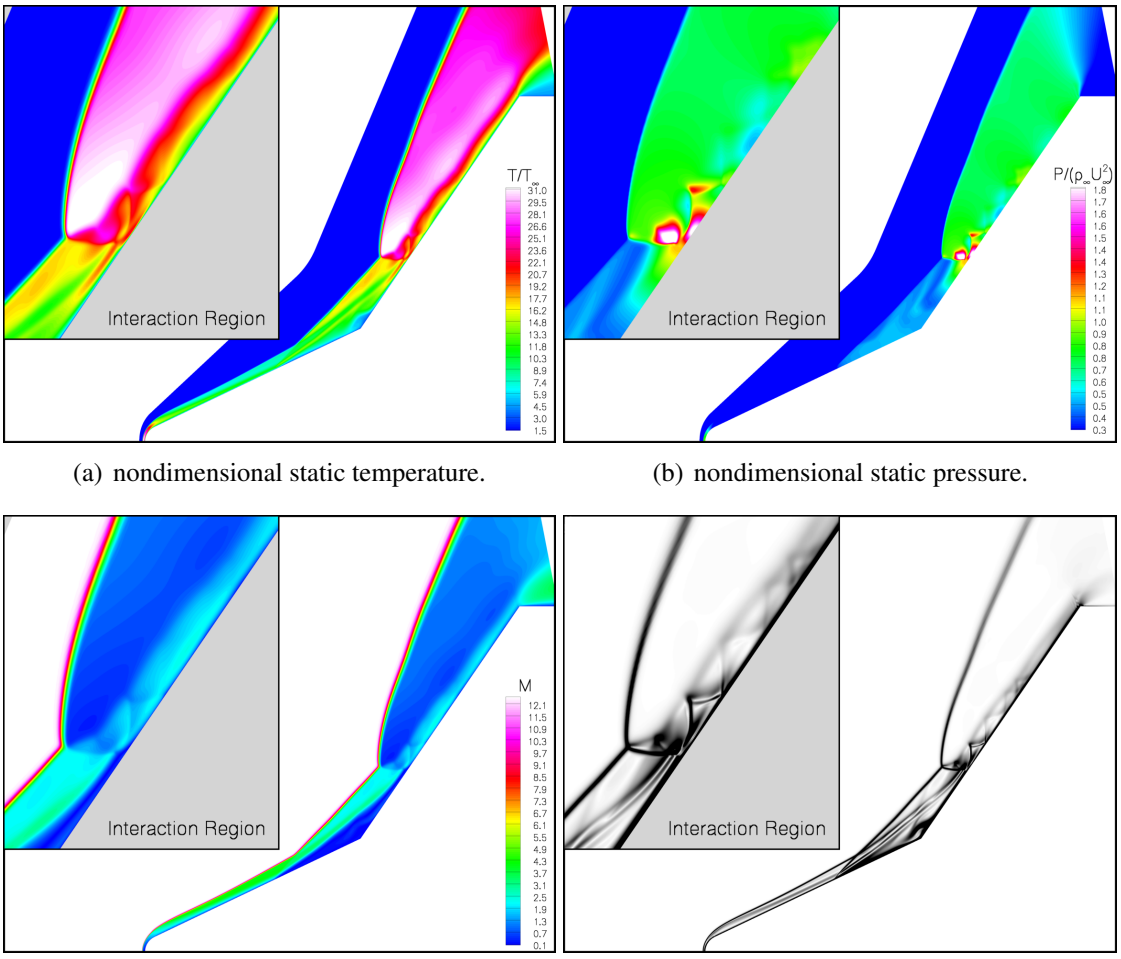


(b) Schematic. Dimensions are inches (millimeters).



(c) Instrumentation Layout.

Figure 5.28: Double cone test article and dimensions [63, 66].



(a) nondimensional static temperature.

(b) nondimensional static pressure.

(c) Mach number.

(d) computed schlieren.

Figure 5.29: Illustration of flowfield for hypersonic blunt double cone shock interaction.

The oblique shock set up by the recirculation region is clearly evident in the figure. This shock overtakes the weak attached shock produced by the 25° forebody and interacts with the detached shock set up by the 55° afterbody. The shock interaction creates a transmitted shock which impacts the model surface, causing a local peak in surface pressure and heat transfer.

The image clearly illustrates the viscous slip surface emanating from the interaction region. This forms a shear layer which separates two distinct regions of flow. Above this feature the flow is subsonic, below it is supersonic. This layer forms the boundary for the complex wave reflection pattern which is observed downstream of the interaction. The transmitted shock reflects from the solid surface as a shock, which is then reflected from the shear layer as an expansion. This pattern continues for several cycles downstream – waves reflecting from solid surfaces as the same type and from the slip surface as opposite type.

Figure 5.30 compares the measured and computed surface pressure and heat transfer for this case. Immediately obvious is the large peak in surface pressure and heat transfer caused by the transmitted shock reflecting from the model surface. Additionally, multiple secondary peaks in surface properties are visible on the afterbody region due to the complex wave reflection discussed previously. It is interesting that the heat transfer and pressure on the forecone upstream of the separation point are not constant (as would be expected for a truly sharp cone). This is in contrast to the sharp cone results which will be considered subsequently. The variations in these properties are due to the phenomenon of "entropy swallowing," which occurs when separate streamlines ingested by the boundary layer contain different entropy. The presence of the blunt nose tip and corresponding curved shock structure induces this behavior.

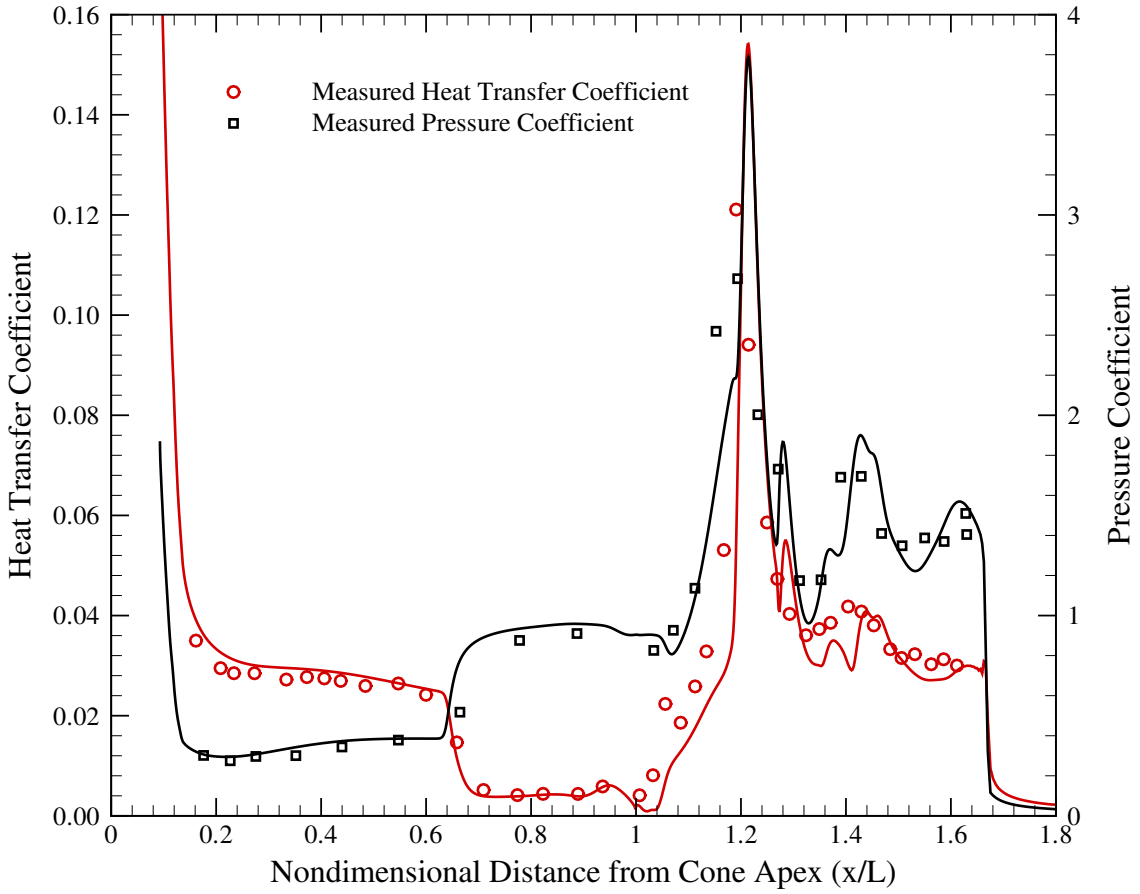


Figure 5.30: Comparison of measured and computed heat transfer and pressure coefficients for cone/cone shock interaction.

5.6.4.2 AEDC Sharp Double Cone

As mentioned previously, the data obtained at AEDC Hypervelocity Wind Tunnel No. 9 and subsequent numerical investigations indicated the effect of vibrational nonequilibrium was negligible. However, these data exhibited considerable unsteadiness for all Reynolds numbers tested. At high Reynolds numbers this behavior is to be expected as the flow will naturally transition from laminar to turbulent, but the observation of unsteadiness for purely laminar flows was unexpected in light of the CUBRC results. Figure 5.31 shows the double cone (with the sharp nose tip) installed in the test section of HVWT9.

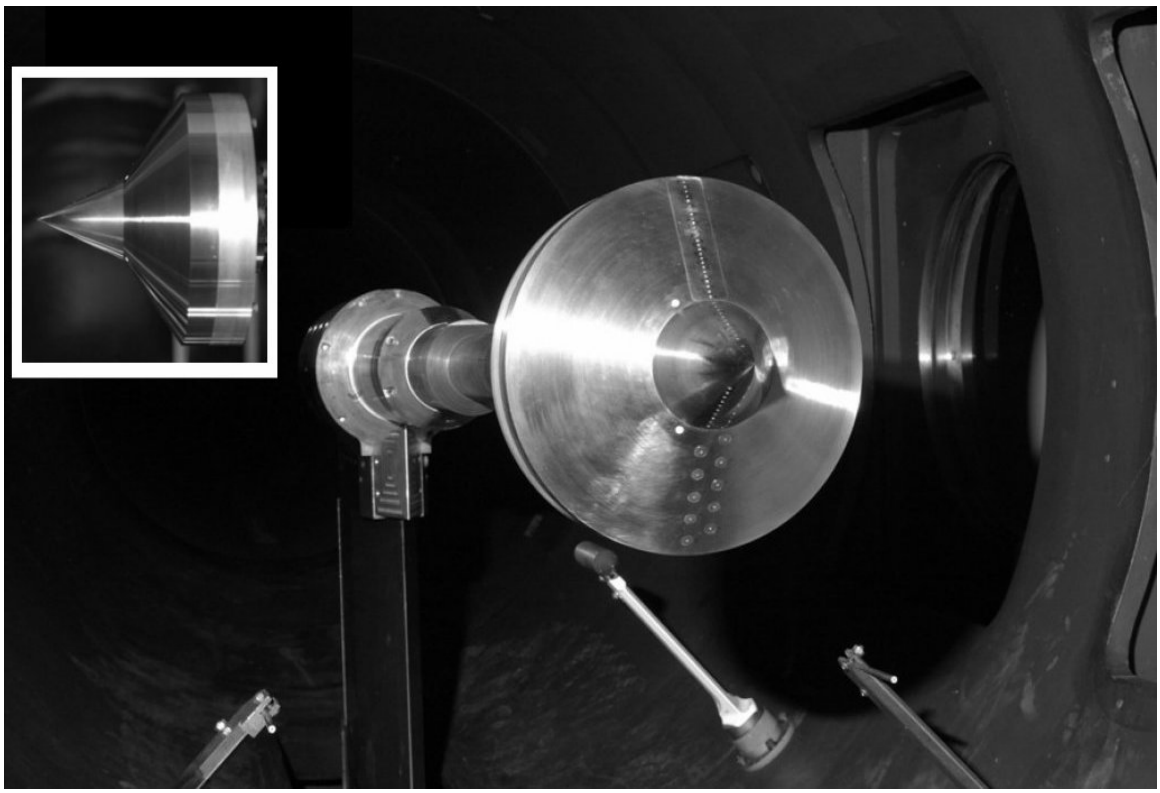


Figure 5.31: Sharp double cone installed in the AEDC Hypervelocity Wind Tunnel No. 9 [65]

The heat transfer and pressure gages are visible on the upper and lower surface of the model, respectively. During the test series surface pressure and heat transfer data

were collected and high-speed schlieren video were acquired. Unfortunately, a light source failure prevented the investigators from obtaining high-speed schlieren for all runs. Surface data were acquired for all runs and were observed to be unsteady. Previous comparisons to computational simulations for fixed freestream conditions showed a substantial discrepancy between the data and the predictions [65].

In this section laminar, calorically perfect Nitrogen flow over the sharp double cone model is examined for all Reynolds numbers tested. First, simulations are conducted with a constant, uniform freestream to determine if the computation predicts a steady-state flowfield. The response of the flowfield to freestream noise is then examined in detail in an attempt to explain the experimentally observed unsteadiness.

5.6.4.2.1 Existence of a steady-state Numerical simulations for the tested conditions were performed assuming a uniform, steady inflow profile based on conditions measured in the facility test section. Even for these steady input profiles, a stable steady-state solution to the Navier–Stokes equations may not exist. This is especially the case for high Reynolds number flows or flows about complex geometries. Time–accurate simulations were performed for all four test conditions assuming fixed, uniform inflow conditions given by the values listed in Table 5.6. It is important that the transient behavior of the flow be modeled accurately, because errors in time may mask the instability of an apparently steady flowfield. In all cases the computational domain was initialized to freestream everywhere and marched in time until either steady-state was achieved or oscillatory behavior was evident.

Runs 2893 and 2894 Using this solution procedure steady-state solutions were obtained for the two lowest Reynolds numbers (runs 2893 and 2894) from the test series. A computed schlieren image for both of these conditions is shown in Figure 5.32. The primary features of the double cone flowfield discussed previously are present for these two cases: a large separated region, a separation shock, and separation shock/aftcone bow shock interaction. Further, the extent of the separated region increases with increasing

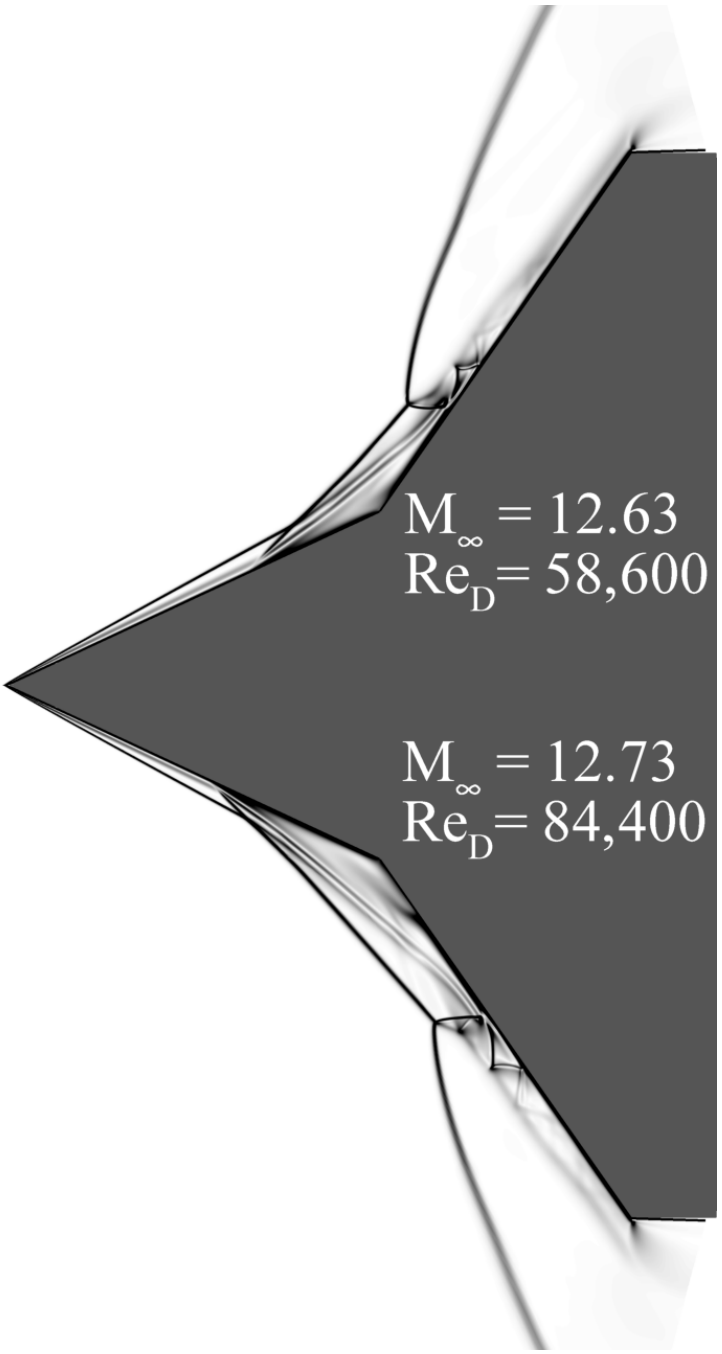


Figure 5.32: Sharp double cone steady-state computed schlieren images for the two lowest Reynolds numbers tested at AEDC Hypervelocity Wind Tunnel No. 9

Table 5.6: AEDC Hypervelocity Wind Tunnel No. 9 sharp double cone freestream conditions [65].

Run	2890	2891	2893	2894	
M_∞	13.6	13.17	12.73	12.63	
Re_D	1.12×10^6	4.11×10^5	8.44×10^4	5.86×10^4	
ρ_∞	7.81×10^{-3}	2.96×10^{-3}	5.90×10^{-4}	3.98×10^{-4}	kg/m^3
U_∞	2006.6	1949.8	1763.5	1682.6	m/sec
T_∞	52.3	52.7	46.1	42.7	K

Reynolds number, as does the width of the compression/expansion reflection layer on the aftcone. With the increase in separated flow size, the strength of recirculation in this region increases, as illustrated by the streamlines plotted along with static temperature in Figure 5.33.

It is clear from this figure that at the lowest Reynolds number tested, $Re_D = 58,600$, the separated region is dominated by one large recirculation zone with a significantly smaller reversed flow region at the $25^\circ/55^\circ$ cone junction. This small recirculation grows substantially as the Reynolds number is increased to $Re_D = 84,400$, as shown in the bottom half of the figure. This corner flow now has a pronounced effect on the primary recirculation region, essentially "pinching" it against the separation shock. The end result is that for the higher Reynolds number case the extent of the separated region and strength of the shock-interaction induced shear layer are both increased. These flow structures are important because it will be shown that, for the case of uniform inflow, it is the interaction of these structures which drives the unsteadiness in the flowfield.

As discussed previously, the two steady-state solutions were obtained by solving the time–accurate Navier–Stokes equations until the time derivative term became negligible.

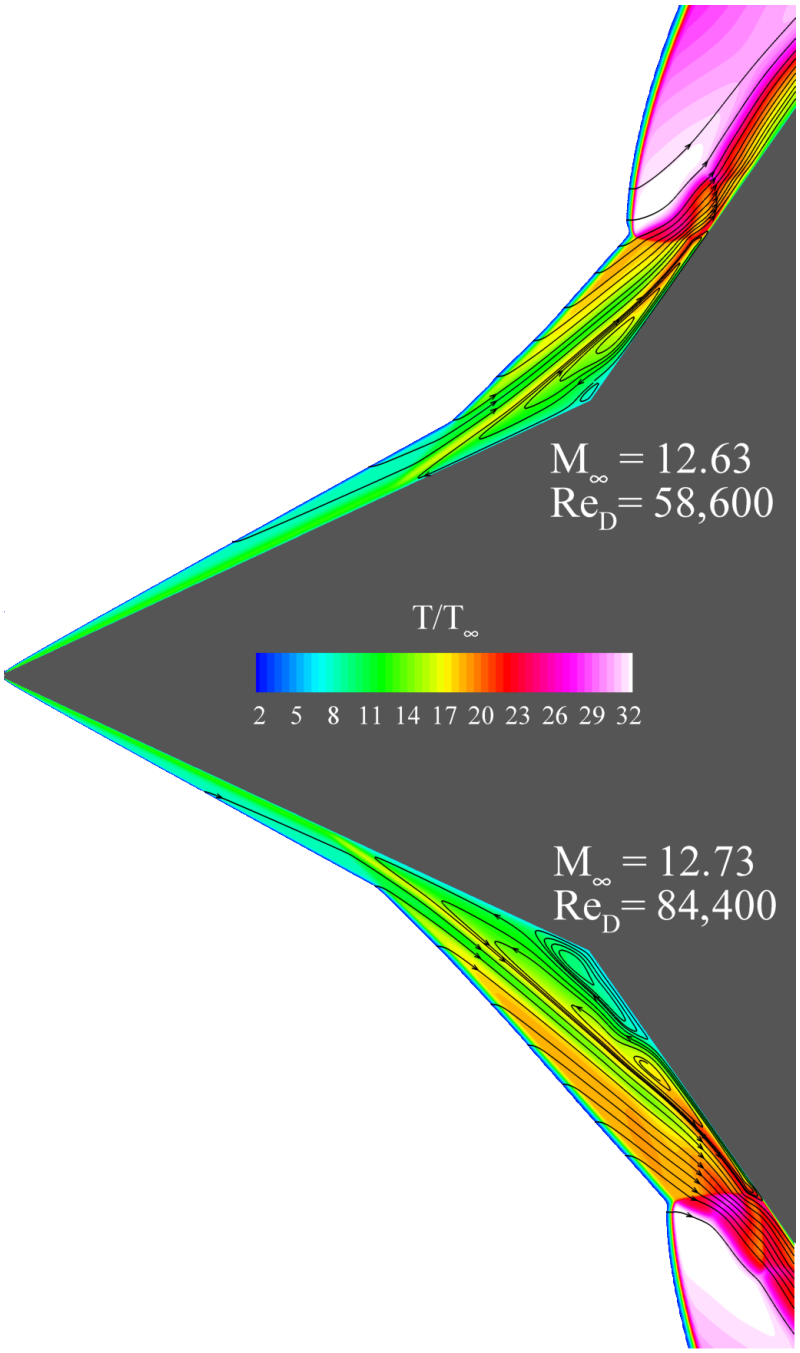


Figure 5.33: Sharp double cone steady-state static temperature and streamlines for the two lowest Reynolds numbers tested at AEDC Hypervelocity Wind Tunnel No. 9

That is, the flow was initialized from freestream values and advanced in a time accurate fashion until $\left|\frac{\partial \boldsymbol{U}}{\partial t}\right| \rightarrow 0$. It is important that the time–accurate discretization be used in this case so that any natural unsteadiness in the flow will be captured accurately and not damped by an overly–dissipative time integration scheme. This approach is in contrast to a typical local time step scheme in which the solution is assumed to be steady a priori, and the integration scheme advances each unknown in the discrete system by some maximum stable time step. The local time step scheme is therefore clearly not time-consistent. This approach is often used to accelerate solution convergence in semi-implicit methods which have some upper bound on the stable time step. This is the case, for example, in the familiar alternating–direction–implicit (ADI) or, to a lesser extent, line relaxation schemes. By contrast, the stability afforded by the fully implicit formulation used in this work allows the solution to be advanced with globally large time steps in a time-consistent fashion.

Figure 5.34 shows the magnitude of the time derivative and the time step size as a function of time step number for run 2894 for a sequence of two meshes (similar results were obtained for run 2893 and are thus omitted). It is interesting that the fully implicit finite element discretization, initialized to freestream values, converges to a steady-state in approximately 250 time steps. This result is especially promising given that the solution is time–accurate and does not resort to an inconsistent time discretization in order to accelerate convergence. The plateau which occurs between time steps 25 and 125 corresponds to the initial development and subsequent growth of the recirculation region. During this time the flow is changing appreciably because as the recirculation grows the separation shock moves, and in turn the shock interaction region moves.

The driving mechanism in this flowfield is the boundary layer separation which occurs due to the adverse pressure gradient induced at the $25^{\circ}/55^{\circ}$ cone junction. The separation region then produces an oblique shock which then impinges on the bow shock created by the aftcone. Given this behavior, obtaining the proper separation point is crucial for simulating the flowfield accurately. To assess mesh convergence, the simulation was performed on a uniformly refined mesh (finer by a factor of four) for run 2894. The baseline

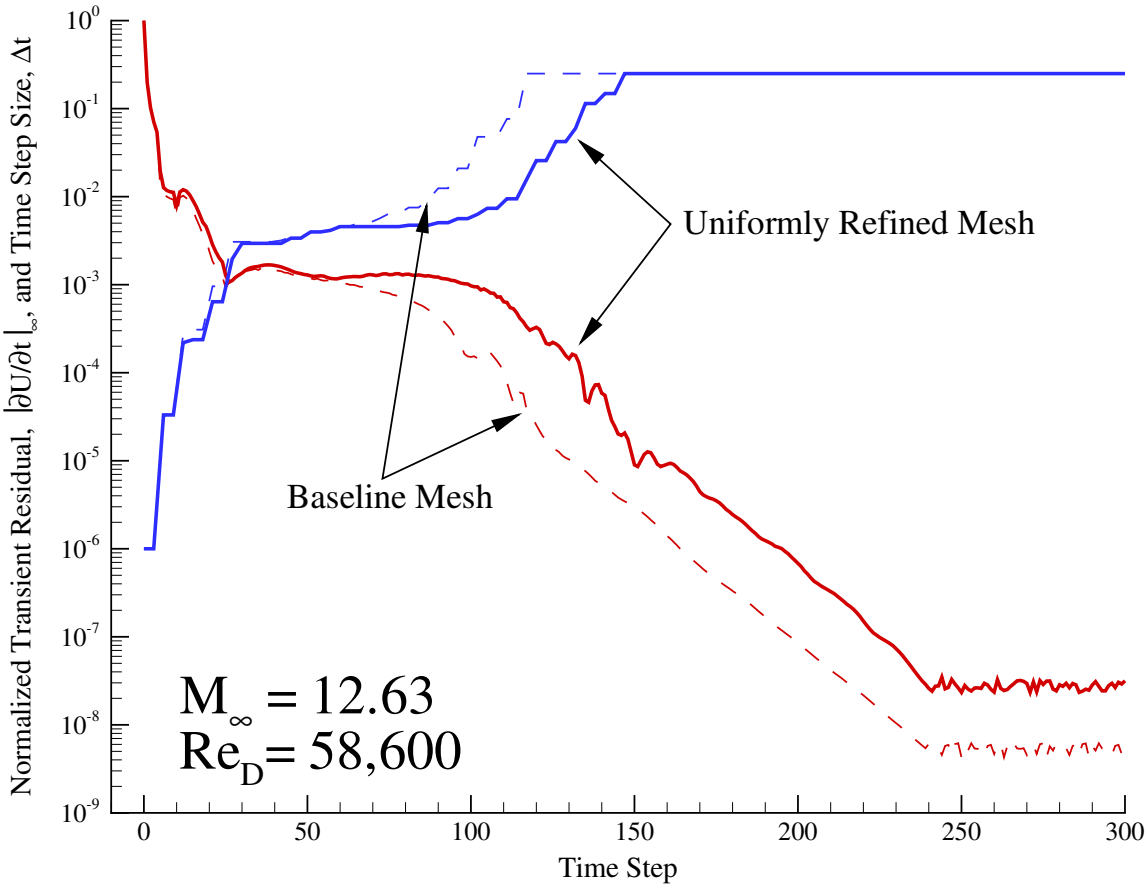
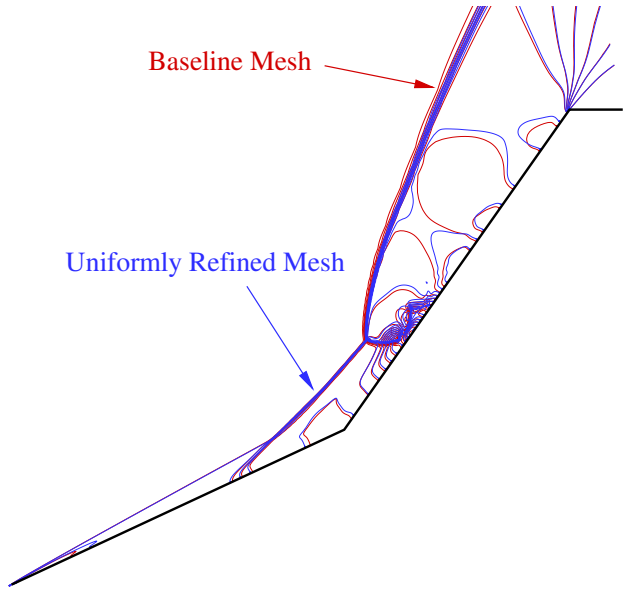


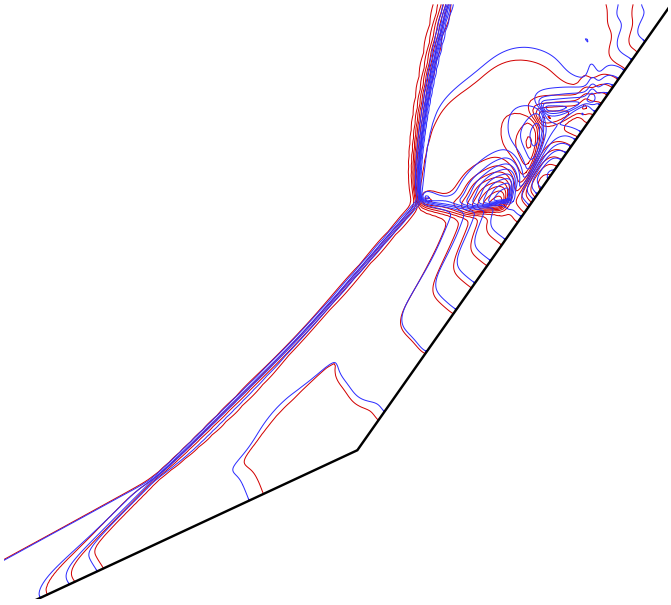
Figure 5.34: Sharp double cone time convergence for run 2894

and fine mesh solutions are compared in Figures 5.35 and 5.36 with overlaid static pressure and temperature contours, respectively.

In general, there is good agreement between the two solutions. The separation, shock interaction, and surface reflection are all captured well. As expected, the shock wave thickness scales with the local mesh size, so the shocks on the uniformly refined mesh are essentially half the width of those on the baseline mesh. It is interesting that the *post* shock location is nearly identical for the two solutions, and it is the shock foot that moves with mesh refinement. This reinforces the observations of Section 5.6.1.

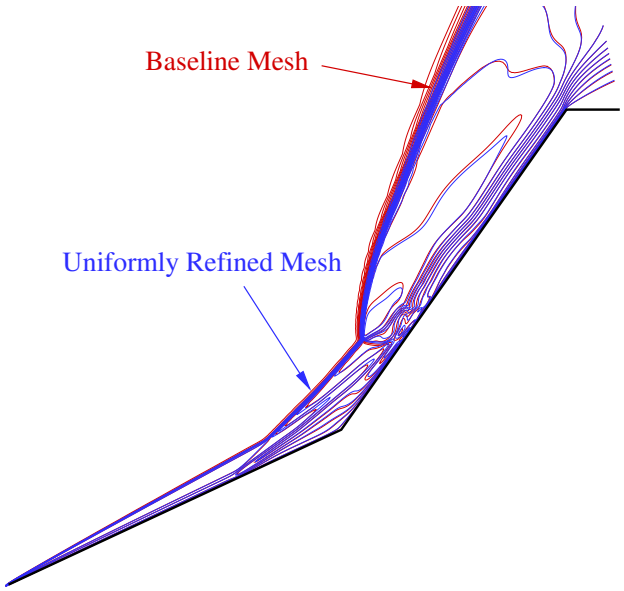


(a) global flowfield

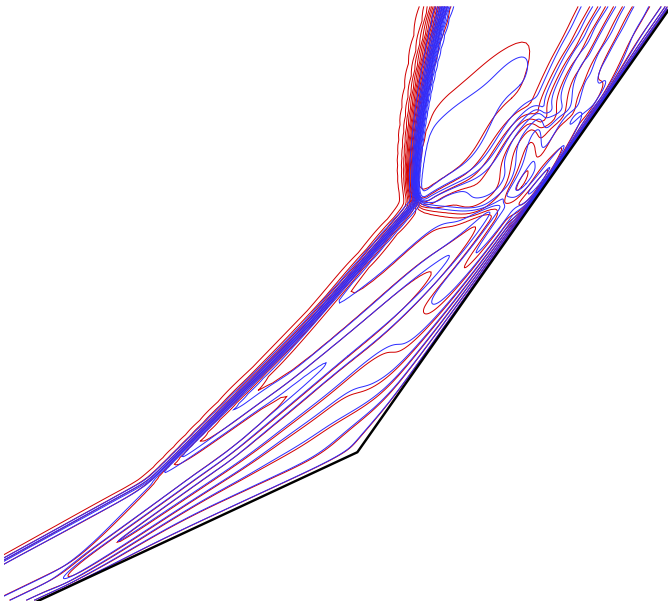


(b) interaction region

Figure 5.35: Sharp double cone – static pressure mesh convergence



(a) global flowfield



(b) interaction region

Figure 5.36: Sharp double cone – static temperature mesh convergence

Runs 2890 and 2891 The same solution procedure was applied to $\mathrm{Re_D} = 411,000$ and 1,120,000, the two highest Reynolds numbers in the test series. A single frame taken from the high-speed schlieren video acquired during run 2890 is shown in Figure 5.37. Given the Reynolds number trend illustrated previously in Figure 5.33, it was expected that the extent of separated flow would be substantially larger and exhibit stronger rotation for these two cases. Indeed, this was found to be the case. For both conditions the separated region contains multiple, counter-rotating vortices which are highly dynamic. This is the case even for a uniform, fixed inflow boundary condition, suggesting that there is no stable, laminar, steady solution to the governing equations for these conditions. While the present work considers an axisymmetric flowfield, it is recognized that the resulting highly dynamic unsteadiness would likely be three-dimensional, even if it were possible to align the model perfectly with a uniform freestream and fix it entirely still. In reality, none of these conditions are possible to meet – the freestream will always contain some nonuniformity, the model cannot be aligned perfectly, and model vibration is present (especially at high Reynolds number and corresponding high freestream dynamic pressure).

Figure 5.38 depicts the instantaneous flow about the model at four points in time for the case of $\mathrm{Re_D} = 411,000$. Clearly, the size and structure of separated flow is highly dynamic and has a strong influence on the resulting downstream shock interaction. Similar behavior was observed for $\mathrm{Re_D} = 1,120,000$ (albeit with a stronger shear layer instability) and is shown in Figure 5.39.

For these two highest Reynolds numbers a Kelvin-Helmholtz instability develops in the shear layer emanating from the shock/shock interaction region. Such an instability occurs when there is a velocity difference between two parallel flows. In the inviscid limit such flows are inherently unstable, while viscous flows may be stabilized by the diffusion of vorticity in the viscous shear layer [68].

In summary, the present numerical method converged to a stable, steady, laminar flowfield for the two lowest Reynolds numbers tested. The two highest Reynolds numbers, however, did not achieve a steady laminar state and contain large-scale dynamic

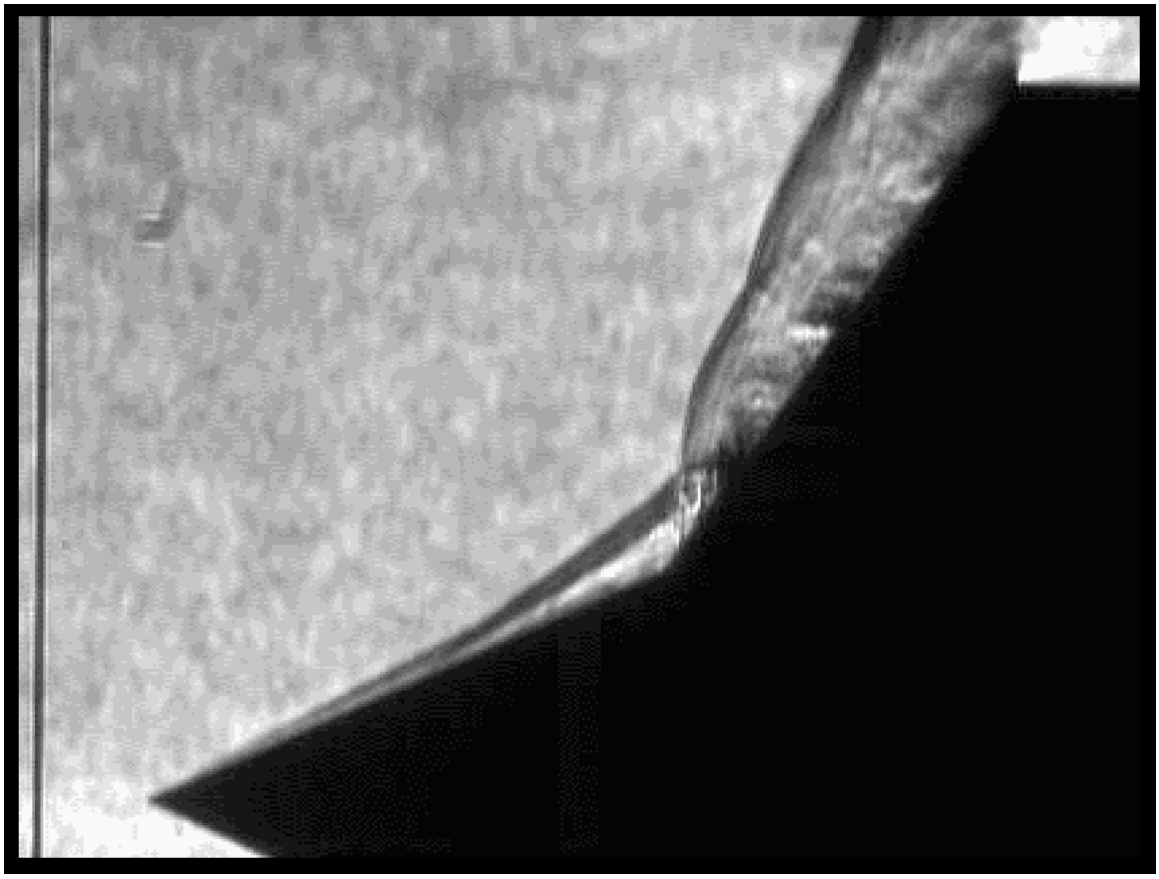


Figure 5.37: Sharp double cone – experimental schlieren image for run 2890.

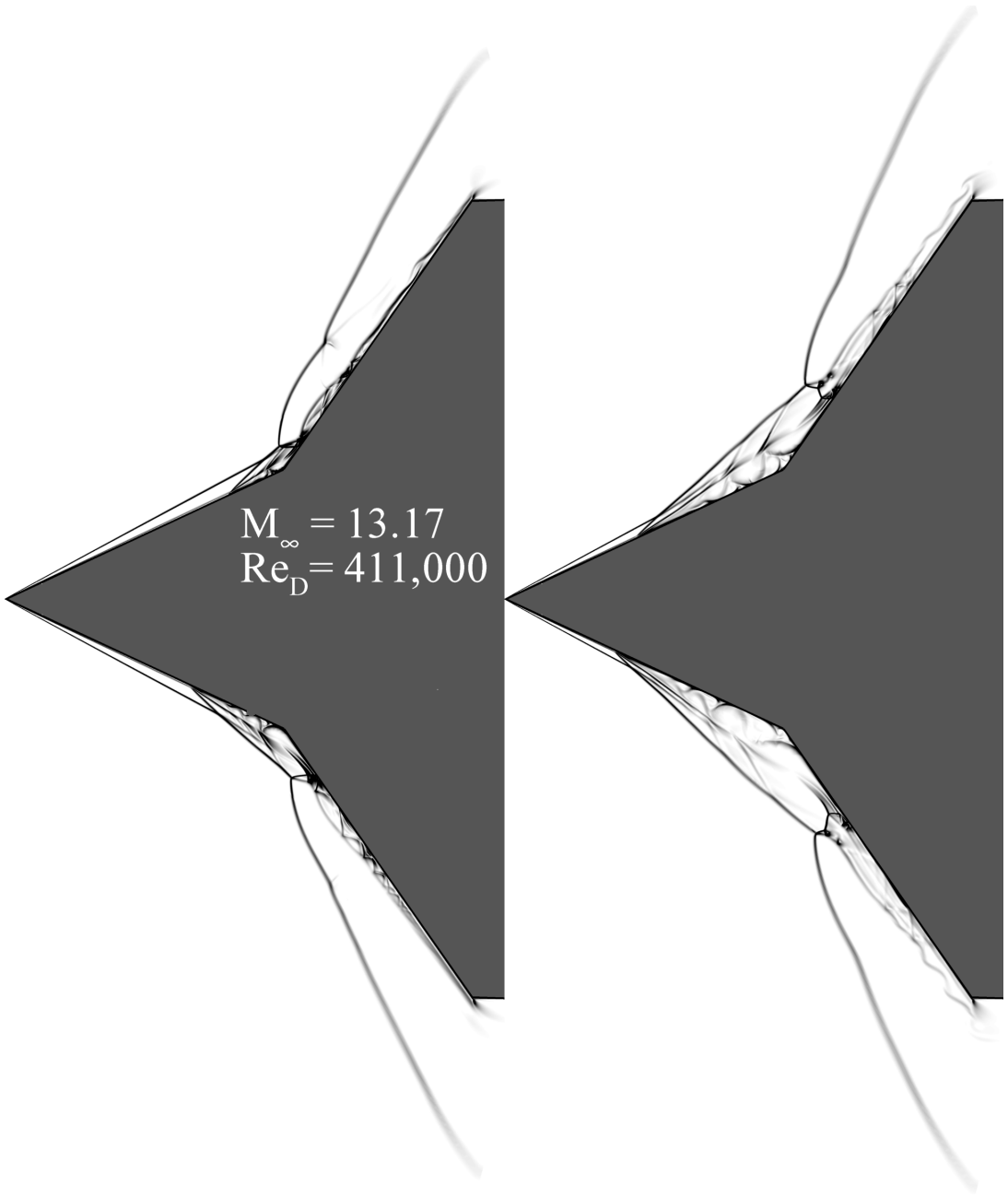


Figure 5.38: Sharp double cone – computed schlieren snapshots at four points in time for run 2891

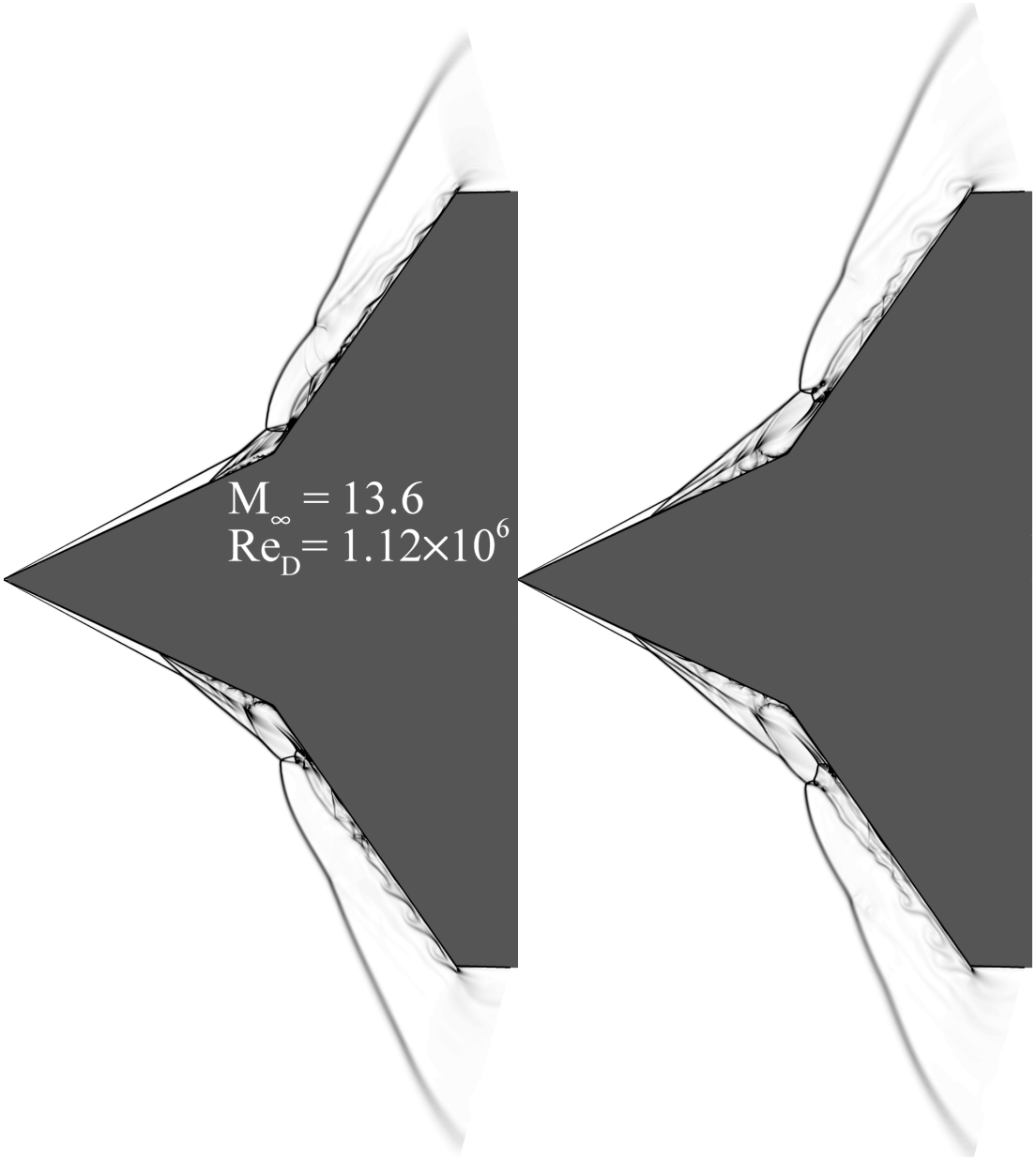


Figure 5.39: Sharp double cone – computed schlieren snapshots at four points in time for run 2890

shock/shock interaction even for the case of uniform inflow. These numerical results are in contrast to the experimental data, which exhibited large scale unsteadiness for all Reynolds numbers tested. The remainder of this section attempts to reconcile these differences first by assessing the impact of the variable model surface temperature on flowfield response and then by modeling the true unsteady freestream which occurs in all conventional wind tunnels.

5.6.4.2.2 Steady state wall temperature sensitivity For all the cases presented previously the cone surface was modeled as a viscous isothermal wall at 300 K. One unique feature of the AEDC tunnel is that, as a blow-down tunnel, it has the ability to produce relatively long run times. Runs in HVWT9 may be on the order of seconds, which is orders of magnitude higher than the order millisecond run times of shock-driven facilities. For the lowest Reynolds number tested, run 2894, the run time was approximately 15 seconds. During this time the model surface may heat up appreciably, especially in regions of localized heat transfer peaks.

The wall temperature variation during the course of a run is of concern because it is expected to have a direct impact on flowfield response. For gases (whose viscosity increases with increasing temperature) wall cooling is known to have a stabilizing effect on the boundary layer. Thus, for a cooler wall, the boundary layer should separate further downstream. As the wall temperature increases, the boundary layer should become less stable and thus separate further upstream on the forecone due to the adverse pressure gradient induced by the base cone. Of course, during the course of a run the model will heat up, so in order to quantify this effect in the context of the experimental investigation the simulation for the lowest Reynolds number was repeated for three different wall temperatures. The wall temperature range of 300–400 K was chosen to bound the experimentally-measured model wall temperature as recorded by the surface thermocouples.

Figure 5.40 shows the results of this parametric study. As discussed previously, the decreasing stability with increasing wall temperature results in a larger separated region.

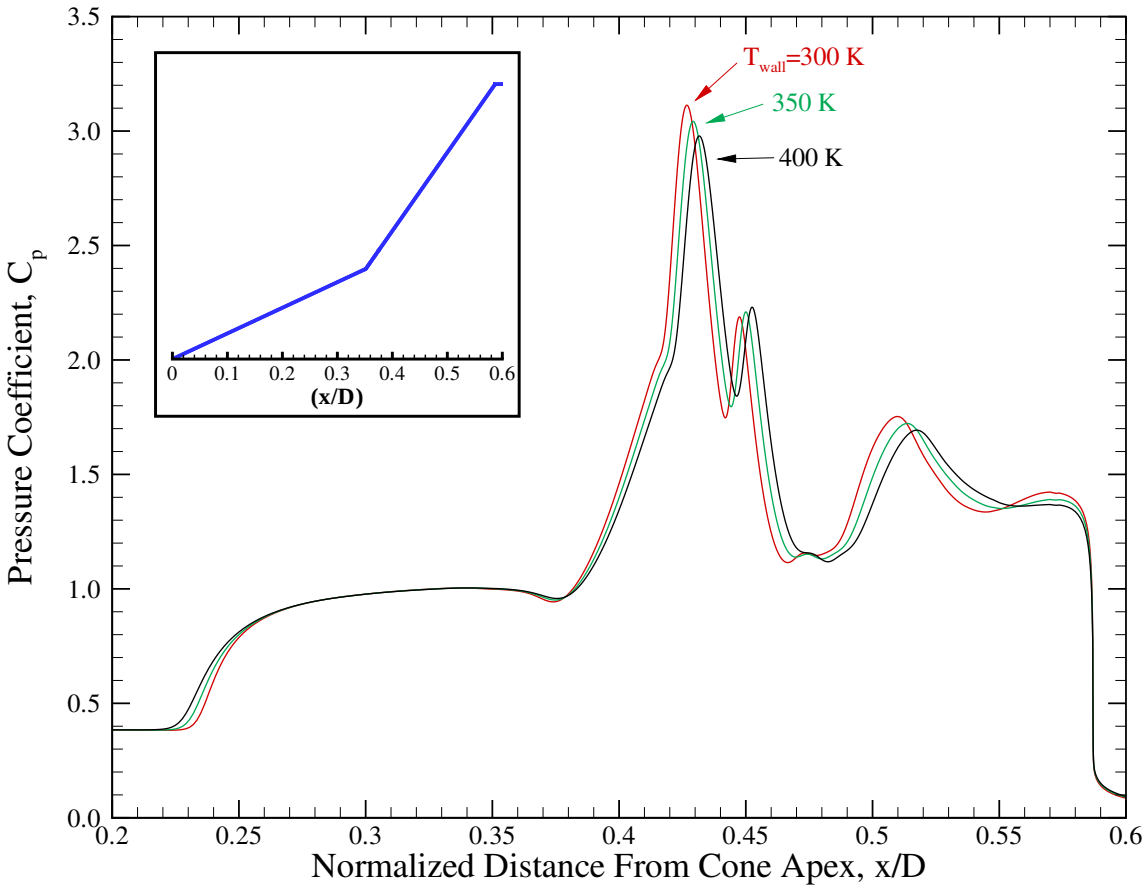


Figure 5.40: Influence of wall temperature on surface pressure distribution for run 2894.

As a result, the overall size of the separated region is found to increase with increasing wall temperature as the upstream separation point moves forward and the downstream reattachment point moves aft. From this study it is clear that, while the wall temperature does influence the size of the separated region, the effect is small for the temperature ranges which could be seen in a typical run. Based on these analyses it is not believed that the experimentally observed unsteadiness is as a result of a temporal variation in the wall temperature.

5.6.4.2.3 Potential source of unsteadiness Conventional wind tunnels such as AEDC Hypervelocity Wind Tunnel No. 9 are known to contain freestream "noise," which may be induced from flow nonuniformities in the nozzle reservoir, freestream turbulence, or from the tunnel wall turbulent boundary layer. In conventional facilities this noise may produce root-mean-squared pitot pressure fluctuations on the order of 1%–5%. These fluctations may produce an unsteady flowfield for a configuration which would otherwise be steady. The remainder of this section examines the freestream noise characteristics of Hypervelocity Wind Tunnel No. 9. In particular, the conjecture that freestream noise is the driving mechanism for the experimentally-observed unsteady flow at the low Reynolds numbers is addressed.

A set of experiments were recently performed in Hypervelocity Wind Tunnel No. 9 specifically designed to characterize freestream noise by measuring pitot acoustic fluctuations. Kulite XT-140 pressure transducers were used to make standard pitot and flush-mounted pitot-acoustic measurements. The measurements provide spectral resolution up to 25 kHz [69]. The test results indicate that for all nozzles used in HVWT9 the freestream noise is on the order of 5% root-mean-squared pitot fluctuation and increases with decreasing Reynolds number. A possible explanation for this trend is that for lower Reynolds numbers the turbulent nozzle boundary layer is thicker and therefore has a larger acoustic influence on the reduced test core.

The power spectral density of the measured fluctuations was also examined. The

power spectral density describes how the power of the signal is distributed with respect to frequency. The tunnel noise was found to be broad and the fluctuations were random with a normal distribution over the measured frequencies of 0–25 kHz [69].

Although the measurements indicate that the freestream noise is broad and randomly distributed over 0–25 kHz, for simplicity it is modeled numerically in this work as a sinusoidal pressure/density fluctuation of a given root-mean-square amplitude at a specified frequency. Additionally, while the measured data were restricted to 25 kHz temporal resolution, the fact that the noise is broad over the range captured suggests there are likely higher frequency components as well, therefore the subsequent numerical experiments will include higher frequencies. Both the freestream density and static pressure are chosen to vary in phase while all other freestream conditions are fixed. Specifically, the applied perturbations are

$$(P, \rho) = \left(\bar{P}, \bar{\rho}\right) \left[1 + \sqrt{2}\Lambda \sin\left(2\pi\lambda t - \frac{x}{\mathrm{U}_\infty}\right)\right] \tag{5.75}$$

where Λ is the root-mean-squared value of the noise perturbation, λ is the specified frequency, and $\bar{()}$ denotes average values. It can be shown from normal shock relations that changes in freestream density and pressure are magnified appreciably through a shock wave; hence, perturbing these properties allows for substantial temporal variation in the flowfield.

Of particular interest is the stability of the shock interaction induced shear layer. At the two highest Reynolds numbers tested, the shear layer was observed to develop a classic Kelvin-Helmholtz instability. This type of instability results when there is a velocity difference between two adjacent layers of fluid. In the inviscid limit, which is approached with increasing Reynolds number, the interface is unstable to all disturbance modes. For finite width shear layers with appreciable vorticity diffusion via viscous forces, however, the interface is subject to the competing effects of diffusion and inviscid layer instability [68]. It is hypothesized that, for the lower Reynolds numbers tested under the assumption of a uniform freestream, this is precisely the case – that viscous diffusion mitigates the natural shear layer instability for the length scales present in the flowfield. However, the question

that remains is whether the shear layer instability (or any other instability, for that matter) will be excited by the freestream noise which is present in the facility.

To analyze the receptivity of the simulation to freestream noise run 2894 was repeated for a range of disturbance frequencies, λ, assuming a representative root-mean-squared pitot pressure fluctuation of 6% [69]. The simulation was initialized with the steady solution previously obtained for uniform inflow. The mean surface pressure coefficient distributions for several specified frequencies are shown in Figure 5.41. The steady

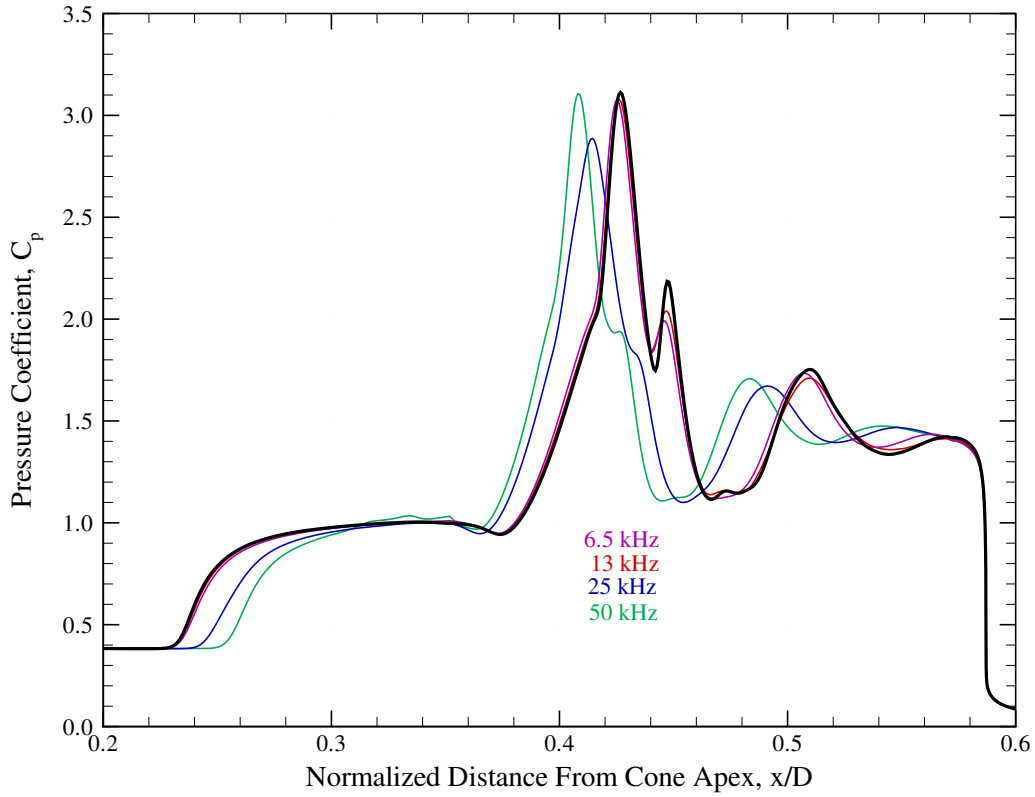
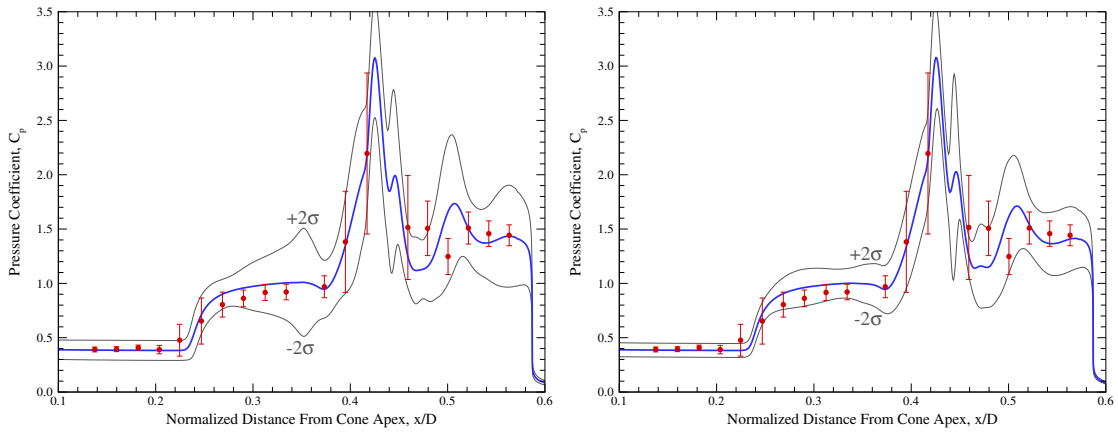


Figure 5.41: Sharp double cone – measured and computed surface pressure coefficient distribution for run 2894 with 6.5–50kHz, 6% RMS freestream noise.

distribution is plotted as the thick black line. There is negligible influence of freestream noise at the two lower frequencies simulated as they produce essentially the same mean surface pressure distribution as the nominal, quiescent freestream case. As the frequency

is increased, however, the extent of the separated region is reduced, suggesting that the additional energy in the oscillating inflow helps delay separation.

Surface traces for these four frequencies are detailed in Figure 5.42. The mean values for both the experimental data and the numerical simulation are computed, as is the standard deviation of the time history for each point. The error bars plotted with the



(a) 6.5 kHz (b) 13 kHz

(c) 25 kHz (d) 50 kHz

Figure 5.42: Sharp double cone – measured and computed surface pressure coefficient distribution for run 2894 with 6% RMS freestream pitot pressure noise for a range of disturbance frequencies.

data correspond to $\pm 2\sigma$, which bound 95% of the experimentally measured values. The

blue solid line corresponds to the average pressure coefficient extracted from the transient simulation. Additionally, the two faint traces bounding the mean correspond to the $\pm 2\sigma$ temporal variation in the computed distributions.

The $\pm 2\sigma$ bounds in the figures illustrate how dynamic the surface interaction is for *all* frequencies simulated. Interestingly, even the lower frequencies exhibit a substantial temporal variance even though the average distribution is not markedly different than for uniform inflow. To illustrate the dynamic nature of the data Figure 5.43 shows instantaneous pressure traces plotted at several discrete times for the case of 50 kHz freestream noise.

Figure 5.44 shows the impact of these freestream fluctuations on the surface heat transfer. Again, the temporal variation in both the data and the predictions are averaged and the standard deviation at each location is computed.

The general flowfield is imaged at four instances for the case of 50 kHz noise in Figure 5.45. There are substantial differences in the structure of the separated region, shock interaction region, and aftcone wave reflection.

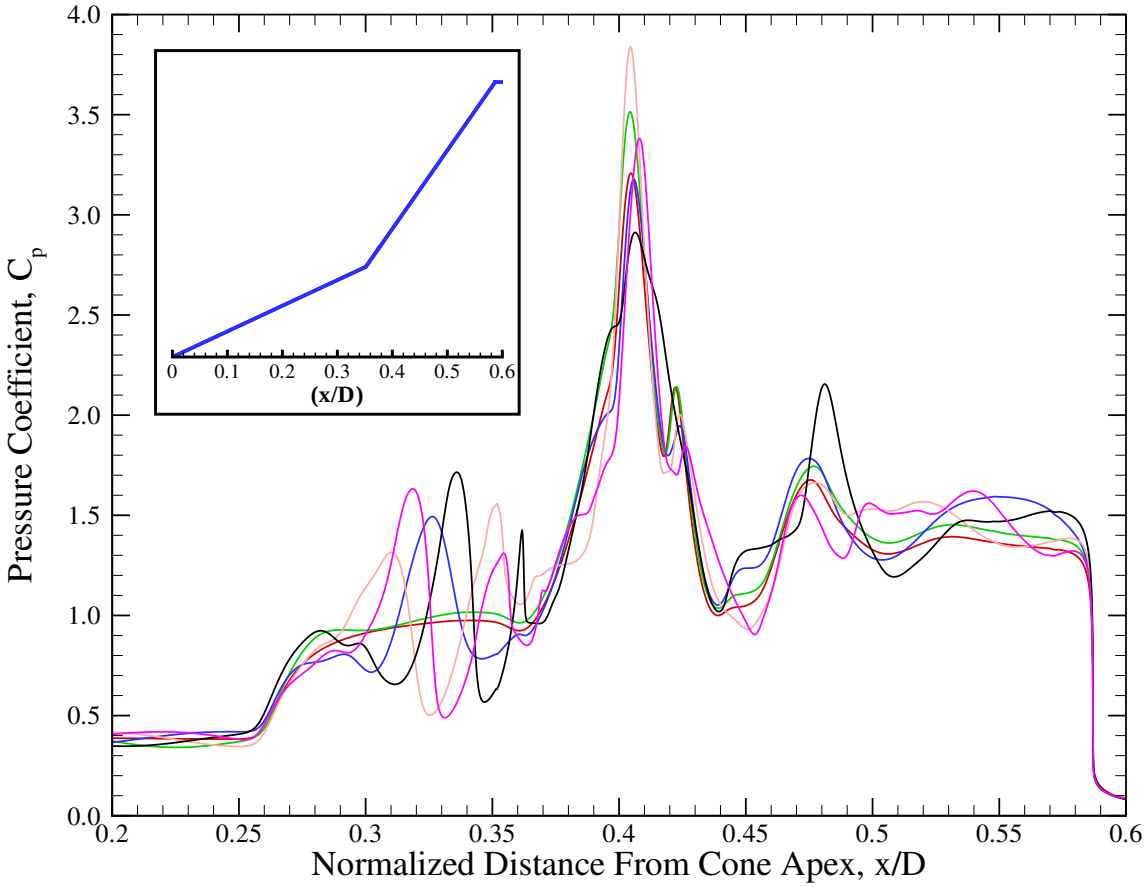
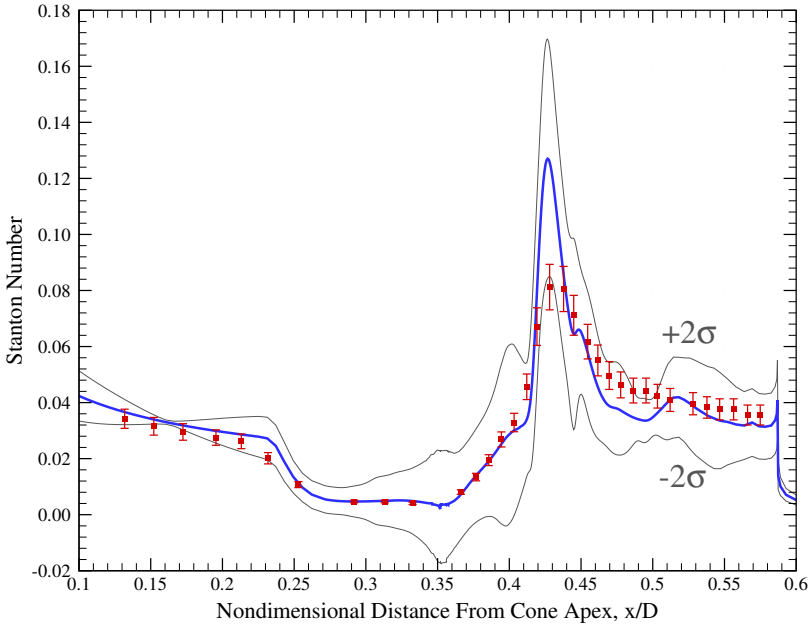
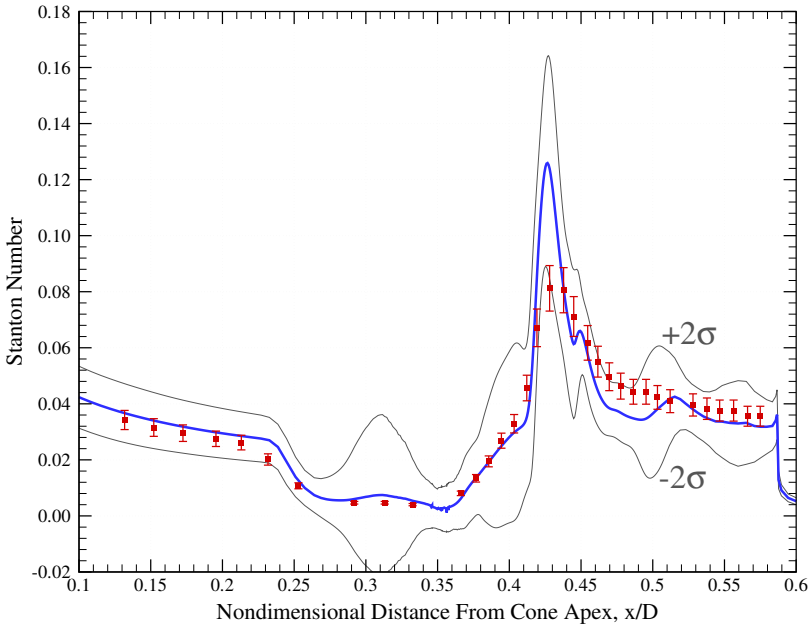


Figure 5.43: Sharp double cone – surface pressure coefficient distribution at several instances for run 2894 with 50kHz, 6% RMS freestream noise.



(a) 25 kHz



(b) 50 kHz

Figure 5.44: Sharp double cone – measured and computed Stanton number distribution for run 2894, 6% RMS freestream noise.

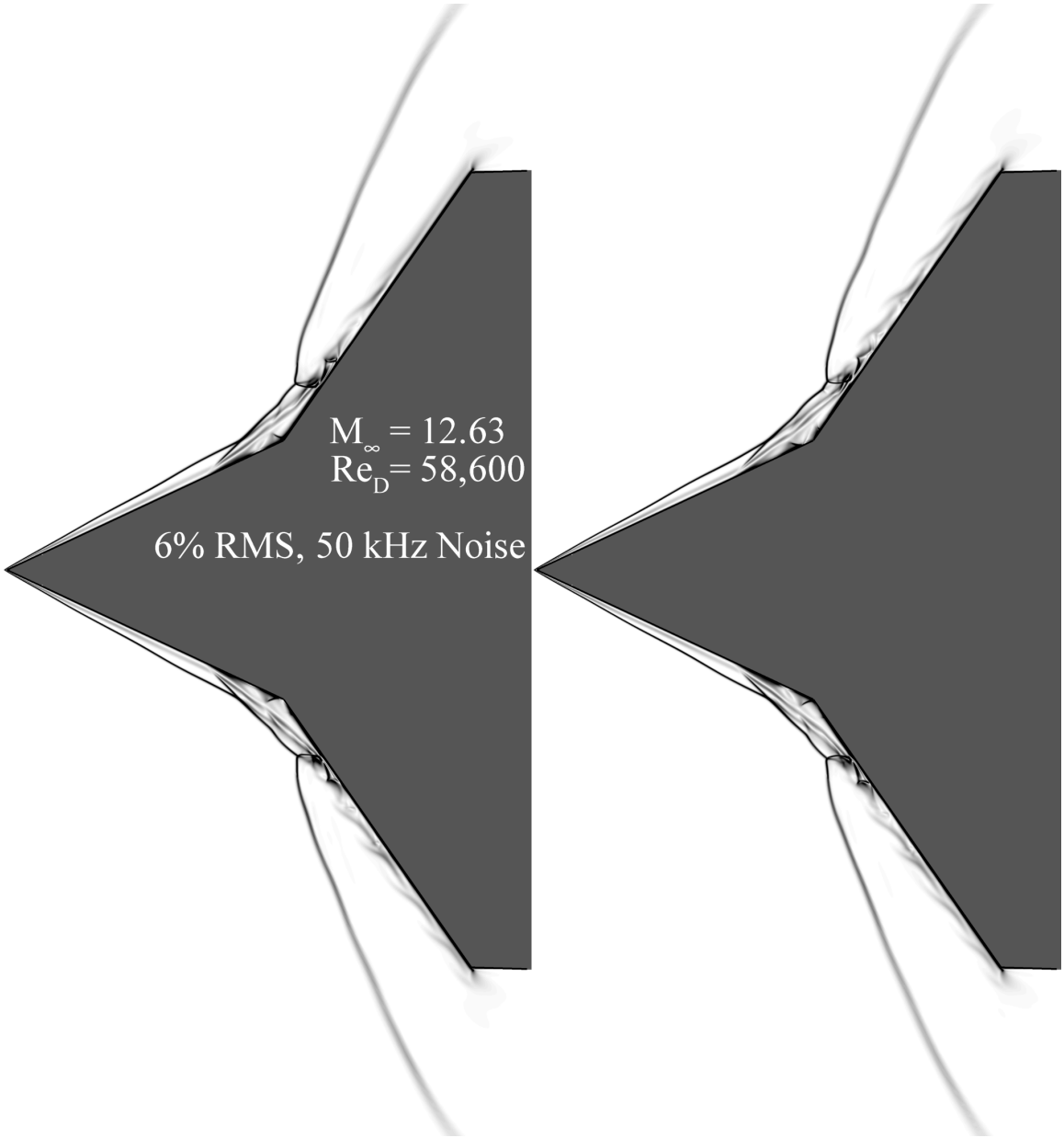


Figure 5.45: Sharp double cone – computed schlieren snapshots at four points in time for run 2894 with 50kHz, 6% RMS freestream noise.

5.6.4.2.4 Observations The experimental data obtained in AEDC Hypervelocity Wind Tunnel No. 9 for four separate Reynolds numbers in the nominally Mach 14 nozzle exhibited large-scale unsteadiness as evident in both high-speed schlieren imagery and surface instrumentation. Unfortunately, the surface gage response time was not adequate to resolve instantaneous heat transfer or pressure distributions, but average values were obtained over long run times.

It was verified for the two lowest Reynolds numbers tested that simulations assuming fixed, uniform inflow converged to steady-state. For the two higher Reynolds numbers the flows were naturally unsteady, predominantly due to the Kelvin-Helmholtz instability in the shear layer emanating from the shock interaction region. The hypothesis that freestream noise is the dominant mechanism for creating the unsteadiness at low Reynolds number was tested by subjecting the lowest Reynolds number case to freestream pitot pressure fluctuations which are representative of the fluctuations seen in HVWT9. The average surface pressure and heat transfer obtained in the presence of freestream noise agree well with the data – although it is difficult to make a conclusive observation due to the lack of temporal resolution in the surface instrumentation. A more rigorous experimental investigation of the influence of freestream noise would be worthwhile and should include high-rate thermocouples or thin-film gages for high temporal resolution of these unsteady flow phenomena.

It is interesting to note that the primary surface response to the sinusoidal freestream perturbation is an in-phase amplitude oscillation of the computed surface quantities of interest. Notably, there is little change in the streamwise location of both the peak heat flux and pressure. Future work could consider unsteadiness in the case of a fully three-dimensional time accurate simulation to investigate the possibility of other unsteady modes which may be precluded by the assumption of axisymmetry. Another possibility for future work is the modeling of the broadband noise present in the facility. This approach will require additional data at various radial stations in the test section to help correlate possible spatial behavior in the noise.

5.6.5 Transient Hypersonic Flow over a Nose Tip/Forward Facing Cavity Configuration

In this section hypersonic flow over a missile nose tip with a forward facing cavity is investigated. This configuration, shown schematically in Figure 5.46, has been observed to exhibit a transient flowfield response in both experimental investigations and numerical simulations [70, 71, 72]. The dynamics of the response are largely driven by the cavity length-to-diameter ratio (L/D). Experimental studies in conventional tunnels report oscillatory response even for relatively shallow cavities, suggesting an unsteady threshold L/D of 0.4. Numerical simulations predict a higher threshold L/D of approximately 1.25 for transient response.

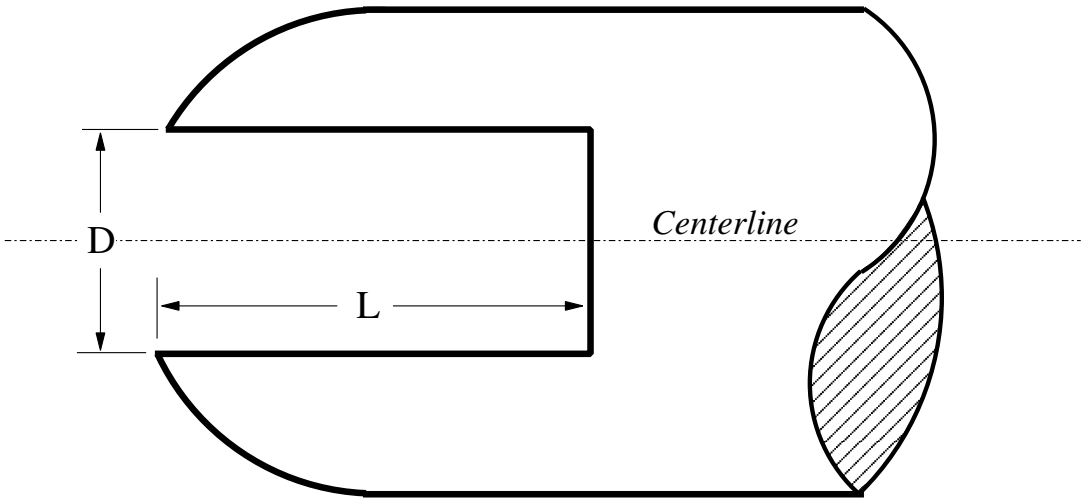


Figure 5.46: Schematic diagram of a forward facing cavity nose tip with a length-to-diameter ratio of 2

Subsequent experiments in a quiet wind tunnel [70] have shown threshold L/D ratios similar to those predicted by numerical experiments, indicating that freestream noise is the mechanism which drives unsteady response in the $0.4 < \text{L/D} < 1.25$ range. Subsequent numerical studies which model freestream noise have verified this observation. Further, numerical studies with relatively deep cavities ($\text{L/D} > 1.5$) exhibit unsteady response even for uniform inflow conditions [71, 72].

The goal of the current analysis is to examine the accuracy and efficiency of the fully implicit finite element scheme for transient applications. Of particular interest is the need to maximize the time step used in the simulation while maintaining time accuracy. It is hypothesized that the fully implicit approach used in this work will admit substantially larger time steps than were possible with the numerical treatments used in previous studies. If this hypothesis is verified the result may reduce the computational time required to simulate a given number of flowfield response periods.

It should be noted that a detailed parametric study of model response for a range of cavity geometries and flow conditions is beyond the scope of this work. (A thorough parametric study was conducted in the experimental setting by Silton [73].) Rather, this example is chosen to test the current algorithm largely *because* of the amount of previous work. This allows the current work to focus on algorithmic details without probing the parametric space of cavity response or attempting to characterize the impact of freestream disturbances.

5.6.5.1 Model Geometry and Flow Conditions

The numerical simulation is chosen to replicate wind tunnel conditions studied by Silton and Goldstein in The University of Texas at Austin's Mach 5 blow-down facility located at the J.J. Pickle Research Campus. In the experimental studies ice models were produced from molds and then cooled in liquid N_2 vapors to a uniform temperature of approximately 78 K. The models were then exposed to the Mach 5 stream and the time required for the onset of melting was recorded. Previous work has focused on coupling the

flow-induced heat flux with the solid body thermal response to predict melting onset time and comparing the predicted and measured values [74].

The model simulated in this case is a spherically blunted cylinder with a nose tip cavity, similar to the notional geometry shown in Figure 5.46. The spherical nose diameter, D_N, is 2.54 cm, the cavity diameter D is 1.031 cm, and the cavity length, L, is 2.062 cm. The lip transition between the cavity wall and the spherical nose segment has a radius of 0.119 cm. The length-to-diameter ratio, L/D, is 2.0, which is sufficiently deep that unsteady motion is expected even for a uniform freestream.

The tunnel was operated at a nominal stagnation pressure, P_0, and temperature, T_0, of 2.3 MPa and 370 K, respectively. The flow is expanded to a freestream Mach number of 4.91, resulting in a freestream unit Reynolds number of $5 \times 10^7 \, {}^1/_{\mathrm{m}}$. This yields a Reynolds number based on model nose diameter, D_N, of $Re_N = 1.2 \times 10^6$. The remaining freestream static properties of interest are presented in Table 5.7. The model wall temperature was fixed at 130 K, while during the run the actual model temperature obviously varied from its initial value of approximately 100 K to 273 K (the model melt temperature of ice).

Table 5.7: Freestream parameters for nose tip/forward facing cavity configuration.

M_∞	Re_N	T_∞	T_w
4.92	1.27×10^6	64. K	130. K

5.6.5.2 Computational Mesh

A block-structured grid topology was used to discretize the domain, as shown in Figure 5.47. The outer boundary of the grid was positioned so as to contain the bow-shock at its farthest upstream location. The off-body grid was created by using a hyperbolic marching technique from the model surface. The primary features of the hyperbolic grid generation scheme used is that it attempts to preserve mesh orthogonality and maintain smooth size transition simultaneously [75]. The mesh contains a total of 28,544 quadrilateral elements with 28,949 nodes.

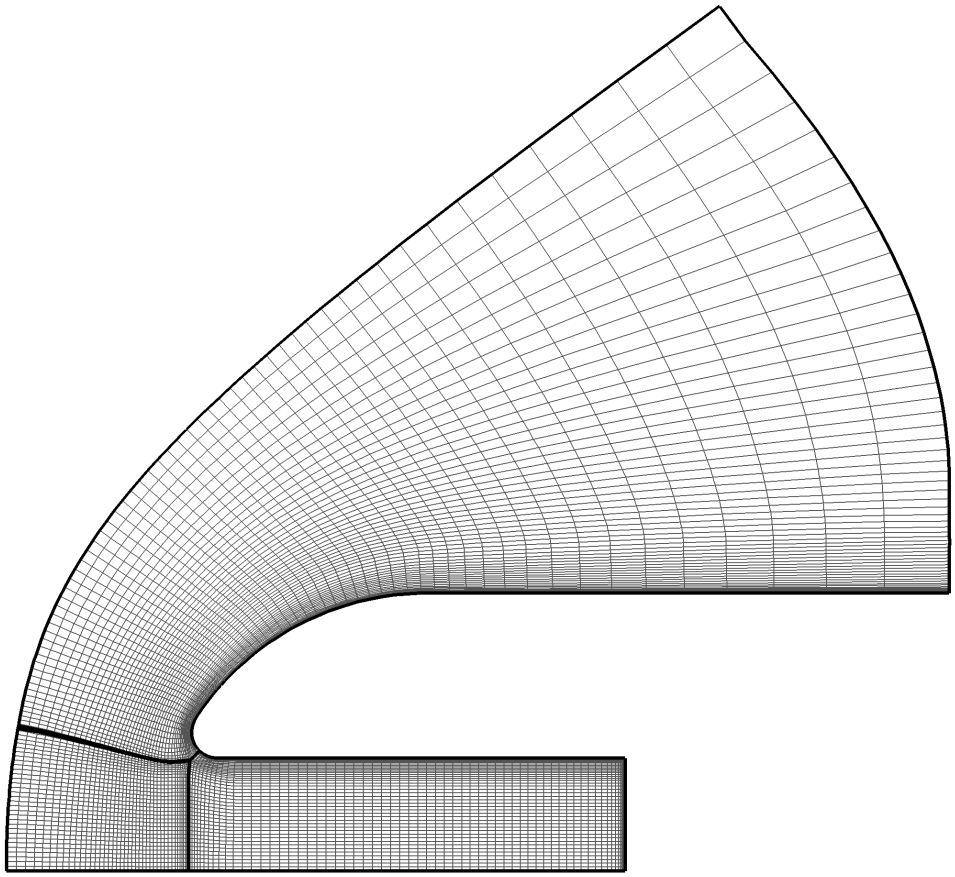


Figure 5.47: Computational mesh used for an L/D=2 cavity in a spherically blunted nose tip. (Every-other point is shown)

The mesh was partitioned into 64 subdomains and run on the *Columbia* supercomputer at NASA Ames Research Center. The partitioned mesh, colored by subdomain, is shown in Figure 5.48. The disparate physical sizes of the subdomains is due to the differences in local element size. Each subdomain contains roughly the same number of elements in order to balance the computational load.

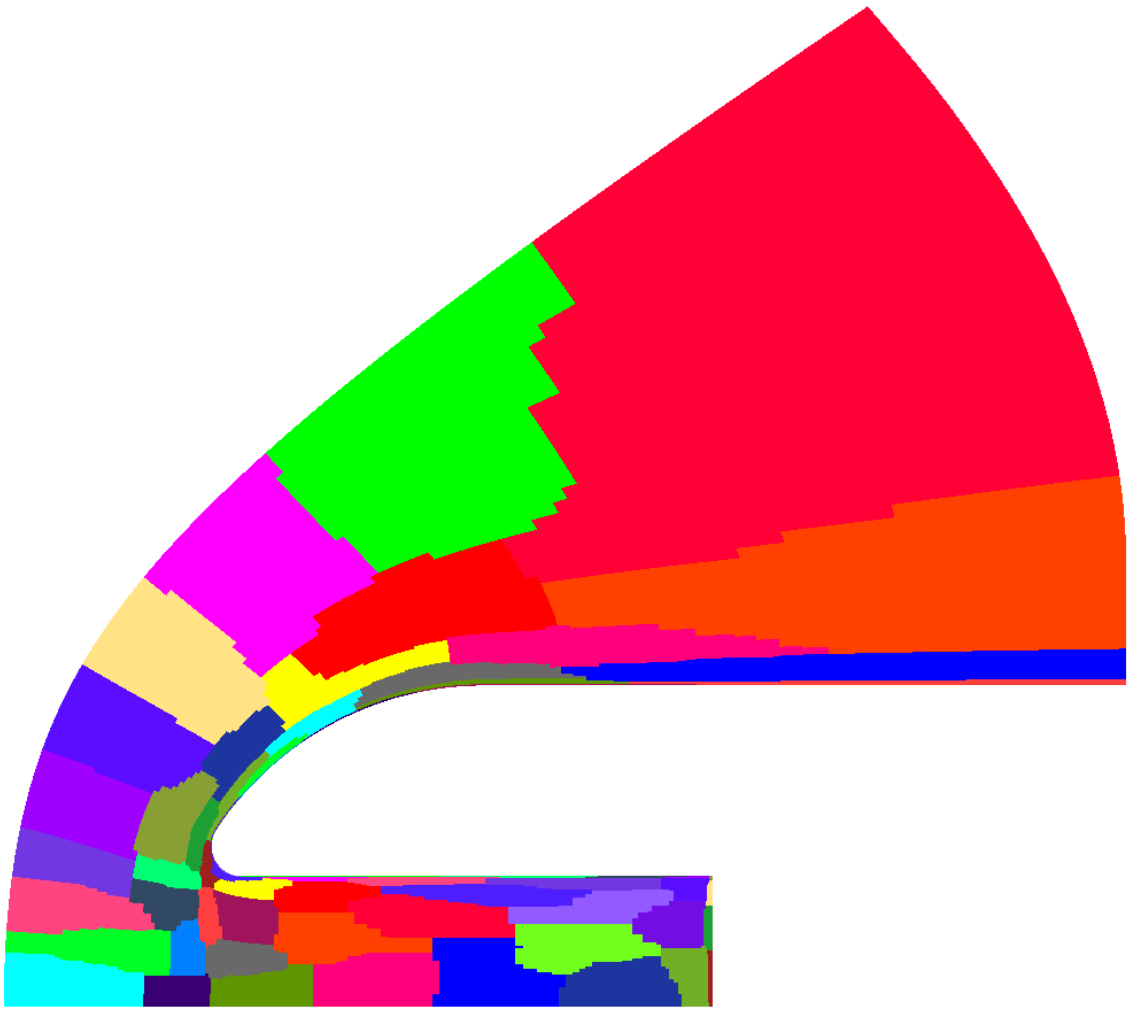


Figure 5.48: Computational mesh used for an L/D=2 cavity in a spherically blunted nose tip partitioned into 64 subdomains.

5.6.5.3 Transient Solution Scheme

The time step control strategy outlined in Section 5.6.1 is modified somewhat here such that the Courant-Friedrichs-Lewy, or CFL, number is the controlled value. The CFL number defines the ratio of characteristic flow length per time step to element size and is often used to describe the time step procedure used in semi-implicit techniques such as the aforementioned LU-SGS algorithm. The CFL-based time step selection used during the simulation is as follows:

1. Compute the characteristic convection time step for each element as

$$\Delta t_e = \frac{h^e_{min}}{\|\boldsymbol{u}\| + c}$$

2. Determine the global minimum characteristic time step

$$\Delta t_{min} = \min(\Delta t_e)$$

3. Set the time step $\Delta t_{n+1} \equiv t_{n+1} - t_n$ for the simulation as a user-specified multiple of Δt_{min}

$$\Delta t_{n+1} = \text{CFL}_{n+1} \times \Delta t_{min} \tag{5.76}$$

where CFL_{n+1} is taken as

$$\text{CFL}_{n+1} = \min\left(\left(\eta^{n+1}\right)\text{CFL}_0, \text{CFL}_{\max}\right), \quad \eta \geq 1, \ n = 0, 1, 2, \ldots \tag{5.77}$$

where η is a user-specified time step growth rate and CFL_0 is the initial time step factor, which is taken as CFL_0=200 in all cases presented in this section.

5.6.5.4 Fixed-Mesh Results

Figure 5.49 shows the static pressure and temperature external to the model and within the cavity at the extreme outboard and inboard bow shock locations. This behavior is consistent with previous research by Engblom and Silton using the commercial CFD code INCA [70, 74]. This cyclic bow shock motion was observed both experimentally and

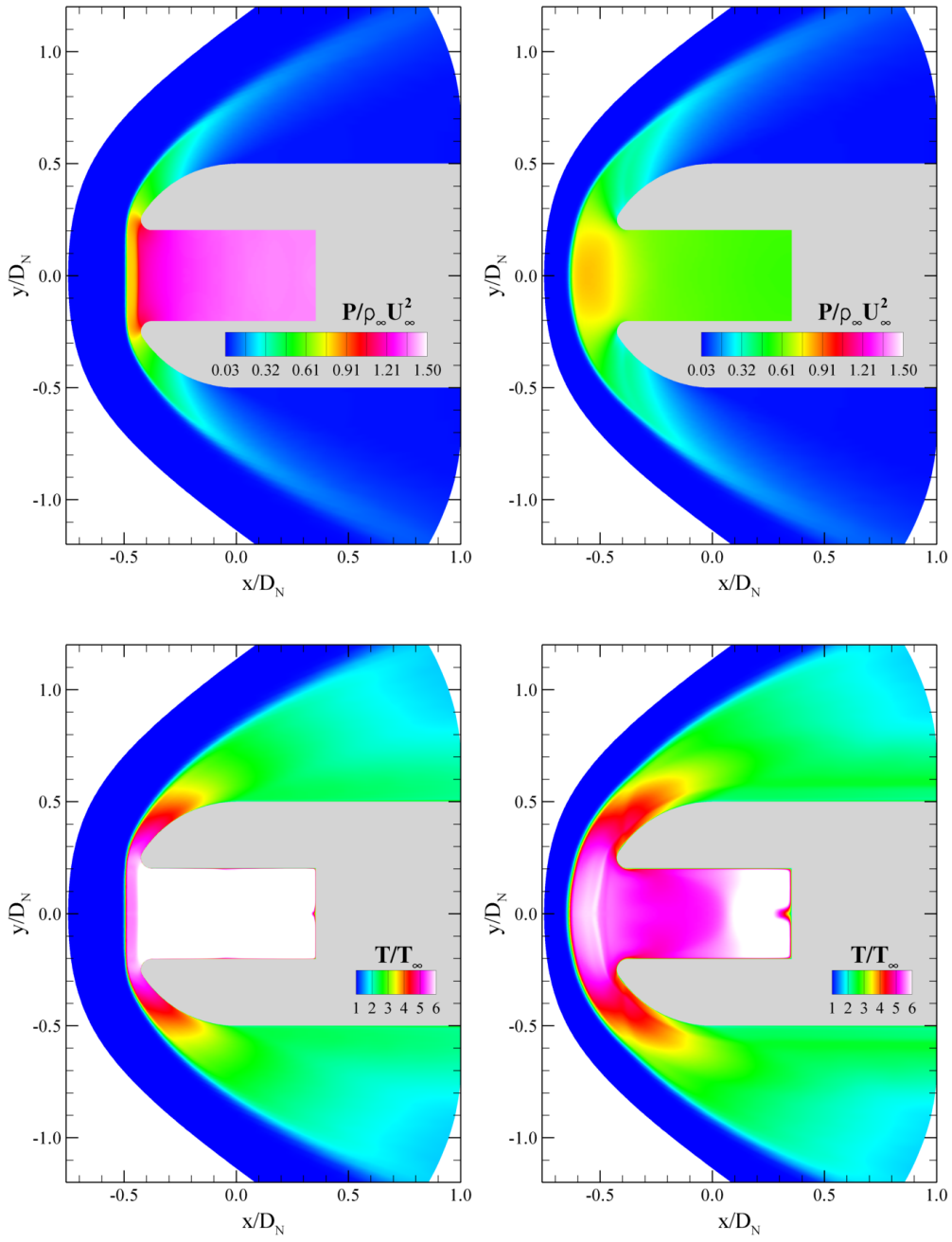


Figure 5.49: Flowfield about a nose tip/forward facing cavity configuration. Static pressure and temperature distribution for bow shock location extrema.

computationally to reduce the net heat transfer into the model. An excellent summary of the fluid dynamic processes which cause this heat flux decrease are presented by Engblom and Goldstein in [71] but merit discussion here as they aid in understanding the nature of the transient flow.

A bow shock oscillation cycle in this problem can be divided into two distinct phases: movement toward the model, and movement away from the model. Assuming that the shock moves in both directions with a representative velocity $\bar{v}_s$, then the shock moving away from the model essentially behaves as a bow shock at a freestream velocity of $U_\infty + \bar{v}_s$. Conversely, when the shock is moving toward the model it behaves as a bow shock in a $U_\infty - \bar{v}_s$ flow. Therefore the stagnation temperature behind the shock (in the model frame of reference) *increases* when the shock is moving away from the model and *decreases* when moving toward the model. In general, the heat transfer to the wall is expected to be proportional to $(T_0 - T_{wall})$, suggesting that the cooler portion of the inflow subcycle would be offset by the hotter portion of the outflow subcycle. Indeed, during the inflow subcycle the relatively cooler flow is attached to the lip of the model; however during the outflow subcycle the relatively hotter flow is convected away from the model surface by flow leaving the cavity. In this way the net heat transfer into the nose tip is reduced.

The primary goal of this analysis was to assess the time accuracy of the current fully implicit finite element algorithm. Previous work using the time–accurate capabilities of the INCA flow solver indicated that on the order of 3000–4000 global time steps with 6 subiterations per time step were required to obtain time-resolved solutions. INCA uses a finite volume formulation coupled with a standard lower–upper symmetric Gauss–Seidel (LU-SGS, see [76, 77]) implicit solver to calculate steady flowfields. The code can perform a number of subcycles per global time step for transient flowfields. By contrast, the fully implicit finite element algorithm used in this work forms the entire linearized system matrix and solves the implicit system with a preconditioned Krylov subspace technique. This approach is considerably more expensive than the LU-SGS scheme in both memory requirements and computational cost per iterate, but has the benefit of stability for sub-

stantially larger time steps. In this way the fully implicit finite element scheme may be competitive or superior to the LU-SGS approach (in terms of wall time per simulation) provided that large enough time steps may be used.

To investigate this possibility, the transient simulation was performed for a range of time steps by sequentially varying $\mathrm{CFL}_{\max}$ in Equation (5.77). In all cases the flow was initiated from freestream conditions and the initial time step was set with CFL_0=200. The time step was then ramped up until $\mathrm{CFL}_{\max}$ was achieved. Four nonlinear iterations were performed at each time step, which generally reduced the nonlinear residual by two orders of magnitude.

Figure 5.50 shows the cavity base pressure vs. time for the series of simulations which were conducted to assess time convergence. Since the point at which time accuracy would degrade was not known a priori an incremental approach was taken in this study in which the time step was varied incrementally. All cases correspond to a total of 12,000 physical time steps with four nonlinear iterations per time step and were run in parallel on 64 processors of the *Columbia* supercomputer [78]. The number of time steps and nonlinear iterations were chosen such that the wall-clock run time would fit within 2 hours.

Results are shown in the figure for maximum CFL numbers of 4, 8, 12, 16, and 20×10^3. The cavity base pressure time history is overlaid for all the simulations to graphically illustrate the time convergence behavior. The long-scale periodic behavior is clearly evident and has a period of $\hat{t} \approx 8$.

According to Engblom et al., this primary 'organ pipe' frequency mode contains the majority of the energy of the oscillation [70]. The frequency of this oscillation is defined as

$$f = \frac{a}{\lambda} \approx \frac{\sqrt{\gamma R T_0}}{4\hat{L}} \tag{5.78}$$

where a is the speed of sound inside the cavity and λ is the characteristic wavelength of oscillation. Assuming that the flow in the cavity is relatively stagnant, the speed of sound can be estimated from the stagnation temperature of the flow. The structure of an oscillation

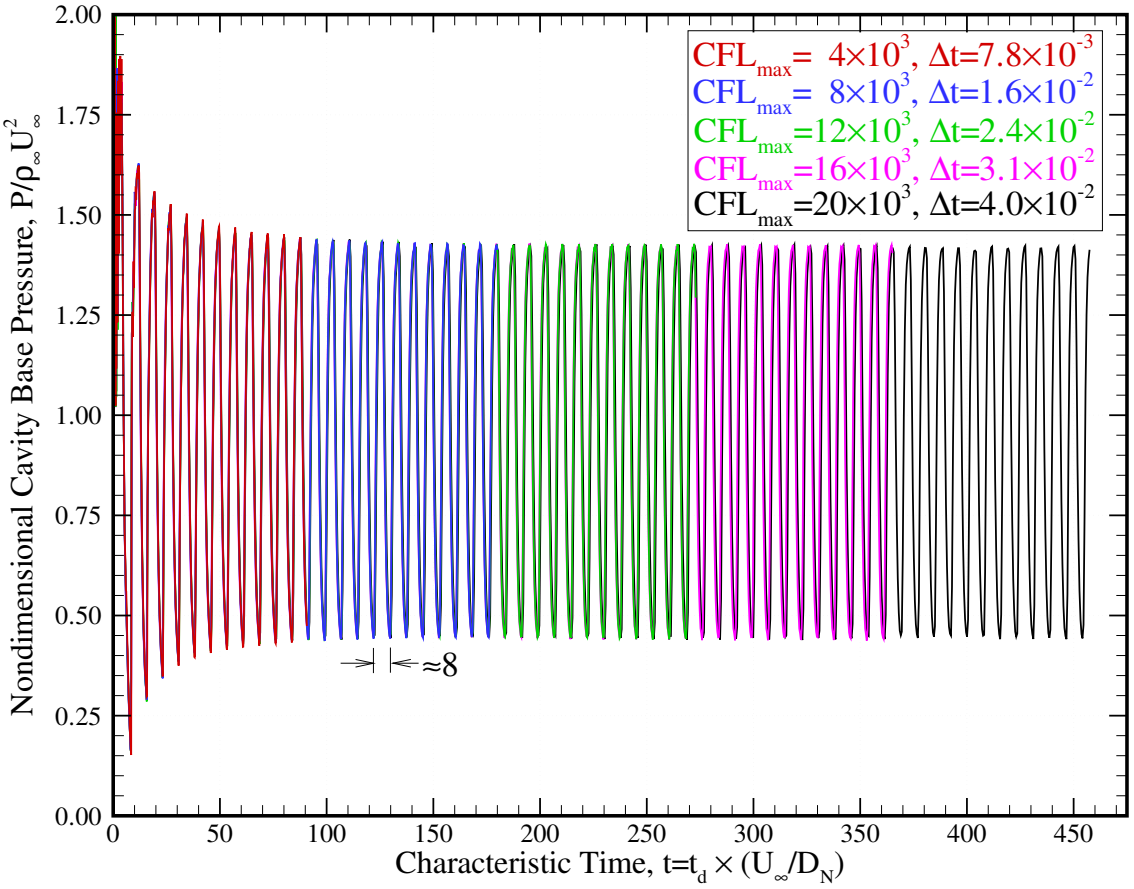


Figure 5.50: Base pressure history for a forward facing cavity with a length-to-diameter ratio of 2

cycle shows that $\lambda = 4\hat{L}$, where $\hat{L}$ is the distance from the base of the cavity to the mean bow shock location along the model centerline. Applying this approximation to the results obtained in this work yields a predicted oscillation frequency of 8.1 (in nondimensional time), which compares well with the oscillation frequency observed in Figure 5.50.

As modeled, the startup phase consists of an initial over-pressure which occurs when the Mach 5 freestream stagnates at the base of the cavity. This over-pressure quickly expands as the flow within the cavity and bow shock set up. The result is an over-expansion, and this cyclic behavior continues for ≈ 50 time units. Previous researchers have observed this behavior and likened it to the response of a damped harmonic oscillator [70, 71].

The initial transient behavior is well captured for all time steps shown in the figure. Since the flow is initialized to freestream values everywhere in the solution domain the startup behavior is rapid and chaotic, and no attempt was made to accurately capture this behavior as it has no impact on the final, periodic state. This somewhat surprising consistency between all five cases helps add confidence in the accuracy and robustness of this time-accurate approach.

For the highest CFL number tested in the numerical experiment the corresponding nondimensional time step was $\Delta t = 4 \times 10^{-2}$. For the observed oscillation period of $\hat{t} = 8$, this results in 200 time steps per oscillation cycle. This corresponds to a factor of 15–20 reduction in the number of time steps per oscillation cycle than the previously applied LU-SGS scheme [71]. It would be interesting to repeat the LU-SGS simulation on the same computer hardware used here so that the the more meaningful comparison of required wall-clock time could be made.

5.6.5.5 Adaptive Mesh Refinement

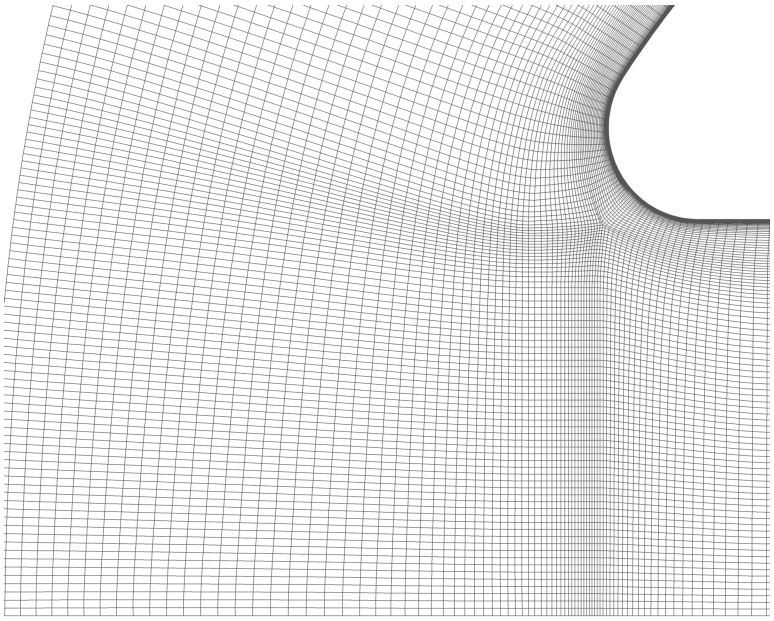
The periodic behavior observed for this case is driven by a series of compression and expansion waves which are alternatively reflected from the cavity base and the bow shock. The fixed cavity base is well-defined as a no-slip surface, however the bow shock

moves rather dramatically during the course of a pressure cycle. Hence, accurately predicting the location of the bow shock throughout the course of the simulation is critical to accurately predicting the periodicity of the response.

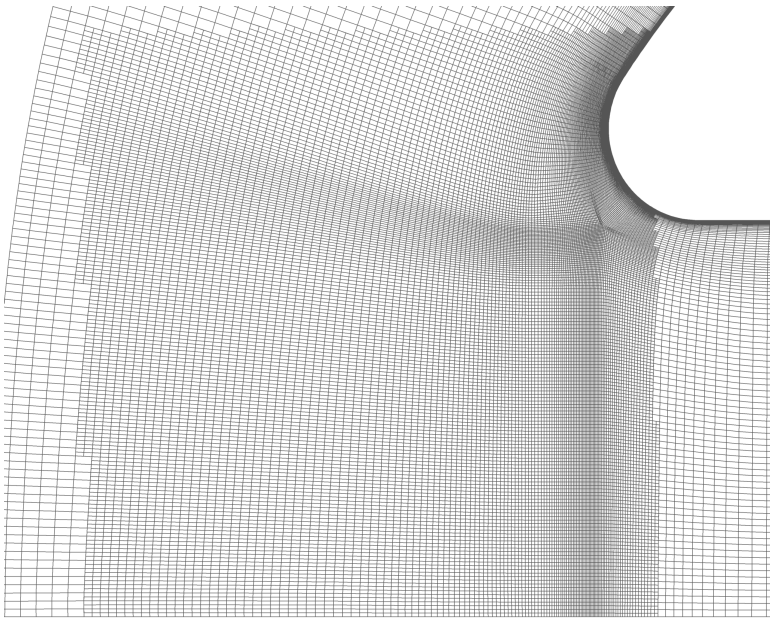
Previous work, which employed block-structured grids, used relatively fine spacing throughout the shock oscillation region to address this issue. The present adaptive finite element scheme admits the alternate possibility of locally adapting the mesh to the shock location to increase spatial resolution.

An incremental approach is used in the case of mesh adaptation. The fixed-mesh results presented previously give insight into the extent of bow shock motion. That information is used to select a subregion of the mesh to uniformly refine to a specified level, and the simulation is repeated on the adapted mesh. This physics-based approach to adaptive mesh refinement is in contrast to the error-estimation approach presented elsewhere in this work, but does illustrates a practical use for the adaptive refinement technology. One obvious argument against this approach is that at any given time step of the problem the mesh is over-refined, particularly in the uniform freestream upstream of the shock. While true, the actual efficiency of this scheme must include the tremendous savings afforded by performing mesh adaptation only *once*, as opposed to at each of the $O(10,000)$ time steps used in the simulation.

This process was performed for two distinct meshes, which are shown in Figures 5.51 and 5.52. For the first mesh all elements whose centroid is located within $-0.7 < x < -0.4$ and $y < 0.3$ were selected for one level of refinement. The resulting mesh is shown in Figure 5.51. This approach refines the region upstream of the cavity which contains the entire range of bow-shock motion. The refinement continues a small distance within the cavity and above the cavity lip. This region was selected for refinement to ensure that the local expansion and separation are captured accurately. The second mesh continues by additionally refining the elements whose centroid is located within $-0.65 < x < -0.45$ and $y < 0.25$. The resulting mesh is shown in Figure 5.51.

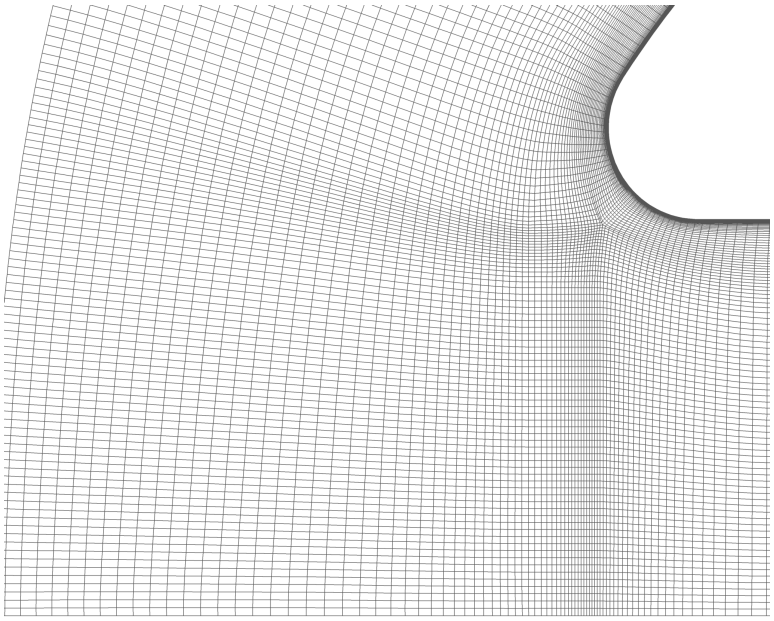


(a) Baseline

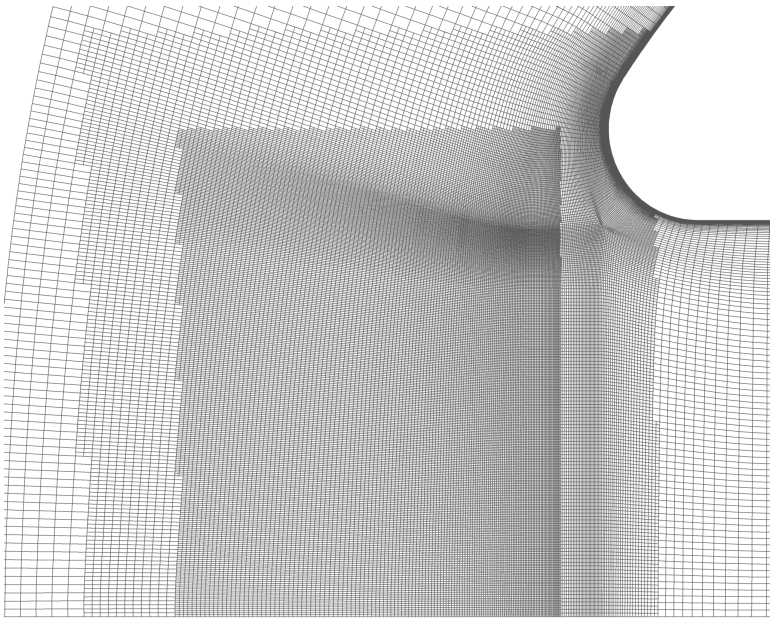


(b) One level of refinement

Figure 5.51: Transient flow about a nose tip/forward facing cavity configuration. Baseline and once-adapted mesh in the lip region.



(a) Baseline



(b) Two levels of refinement

Figure 5.52: Transient flow about a nose tip/forward facing cavity configuration. Baseline and twice-adapted mesh in the lip region.

The motivation for this study was that the bow shock location is critical for predicting the proper periodicity in the flowfield, as it is reflection off this bow shock that completes the pressure oscillation cycle. To test the adequacy of the baseline mesh the simulation was repeated on these two meshes and the cavity base pressure was again extracted as a function of time. These pressure histories were compared to that of the baseline mesh and found to be essentially identical. Thus, they are omitted from the presentation of results. It is clear from this study, however, that the baseline mesh is adequate for resolving the dominant unsteady mode of this flowfield.

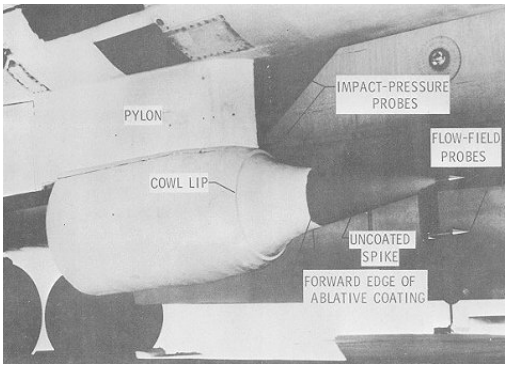
5.6.6 Shock-Shock Interaction

Shock–shock interaction (SSI) is a physical phenomenon that occurs when multiple shock waves in a compressible flowfield intersect and interact in some way. When such interactions occur, very complicated and localized aerothermodynamic processes may result which can adversely impact a supersonic vehicle by creating regions of intense heating (easily an order of magnitude higher than in the case of undisturbed flow) or extremely high pressure loads. In general, the intensity of these interactions becomes more severe in hypersonic flight regimes. Additionally, analysis of SSI effects on a vehicle that operates in the hypersonic regime can be complicated by the presence of real gas effects. This section considers additional shock interaction patterns beyond those observed in the previous sections.

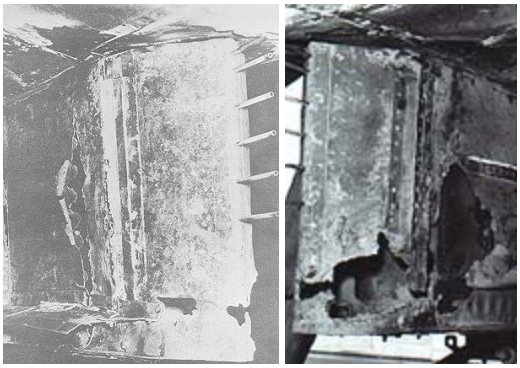
Historical examples of hypersonic flight vehicles such as the X-15 rocket plane and Space Shuttle Orbiter indicate that the consequences of not accounting for SSI effects in vehicle design can be severe. Indeed, in hypersonic flight regimes SSI effects can easily cause complete vehicle failure if not accounted for in the vehicle design phase. A classic example of this was the X-15 flight experiment in which a dummy ramjet was hung from the lower surface of the vehicle (shown in Figure 5.53). The resulting ramjet/pylon shock interaction produced local heat fluxes approximately 7 times nominal values, which resulted in failure of the ramjet attachment as shown in Figure 5.53(c) [79]. As modern trends in aerospace technology focus on reusable, efficient hypersonic vehicles the role SSI effects play in vehicle design becomes even more pronounced.

A literature review illustrates that little thought was given to the SSI interaction problem until the end of the 1960's. Since that time a significant amount of work has led to a basic understanding of the problem that will presented in the following sections. However, what is interesting about the SSI phenomenon is that when compiling a list of factors important in the accurate prediction of SSI effects, we find that our list includes nearly all of the current research phenomena in high speed fluid dynamics. These phenomena include (but are certainly not limited to):

(a) Separating from the B-52



(b) Pre-flight



(c) Post-flight

Figure 5.53: Shock–Shock interaction during the X-15 dummy ramjet flight experiment. During this flight the X-15 accelerated to a freestream Mach number of 6.72 [79].

- Real gas effects in hypersonic flight regimes
- Shear layer/boundary layer instability and transition
- Shock wave/boundary layer interaction
- Adequate shock wave resolution in computational models
- Accurate convective heat transfer rate measurement and prediction

The extensive work by Edney [80] was the first in–depth experimental research that considered the effect of various shock interaction patterns on a number of geometries. Edney examined a number of glass models in a Mach 4.6 wind tunnel. The models were instrumented with platinum thin–film gages that were used to measure heat transfer rates. These gages were connected to an analog circuit network that allowed Edney to calibrate and measure the surface heat flux. An oscilloscope was used to visualize the heating distribution around the glass models that these gages recorded. Static ports and a quasi–static injection technique was used to measure the static pressure distribution on the model surface.

The general setup Edney used is illustrated in Figure 5.54. The flow direction in the figure is from left to right. A flat plate with a slot cut through it was placed in the test section of the wind tunnel. The tunnel was started and an oblique shock of known strength developed from the leading edge of this plate. The model, which was mounted on a sting, was then injected into the flowfield through the slot in the flat plate. As the model entered the flowfield it passed through the boundary layer associated with the tunnel wall, through the plate boundary layer, and finally came to rest in such a way that the oblique shock formed by the flat plate interacted with the bow shock created by the model. In this way Edney was able to accurately measure the heat flux and pressure loads on the surface of the model that resulted. Edney was able to adjust the plate angle to obtain oblique shocks of various strengths and angles. He tested hemispheres, cylinders, and wedge–shaped models in [80].

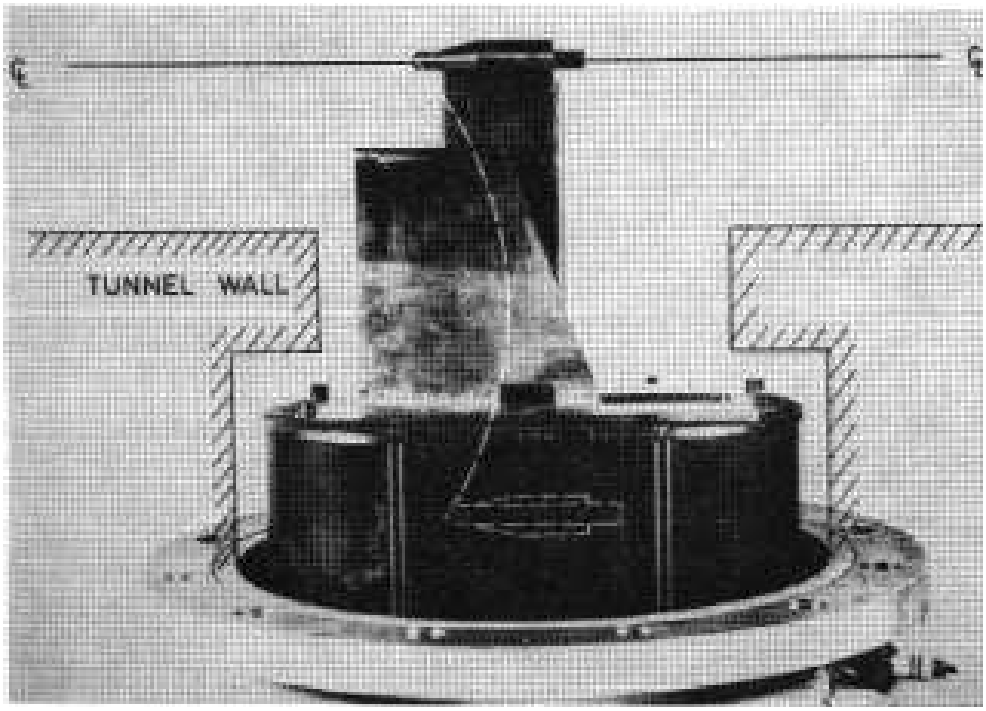


Figure 5.54: Edney's experimental setup used to study two-dimensional shock-shock interaction [80]

Edney identified six distinct interaction patterns that result from SSI. He also identified four key factors (for laminar interactions) that dictate the pattern that results, including: (i) freestream Mach number, (ii) geometry, (iii) strength of the impinging shock, and (iv) location of the impinging shock. These distinct patterns are commonly noted as type I–type VI shock interactions. The type VI interaction with its resulting shear layer was the dominant feature in the previous double cone simulations. Two additional types, the type IV and VI will be considered here. The type IV interaction as it produces the largest local augmentation to surface pressure and heat transfer of all types. The type VI interaction is of interest because such a pattern results from the bow shock/wing shock interaction for the Space Shuttle Orbiter at reentry attitude.

Edney also developed analytical methods of analyzing these various interaction pat-

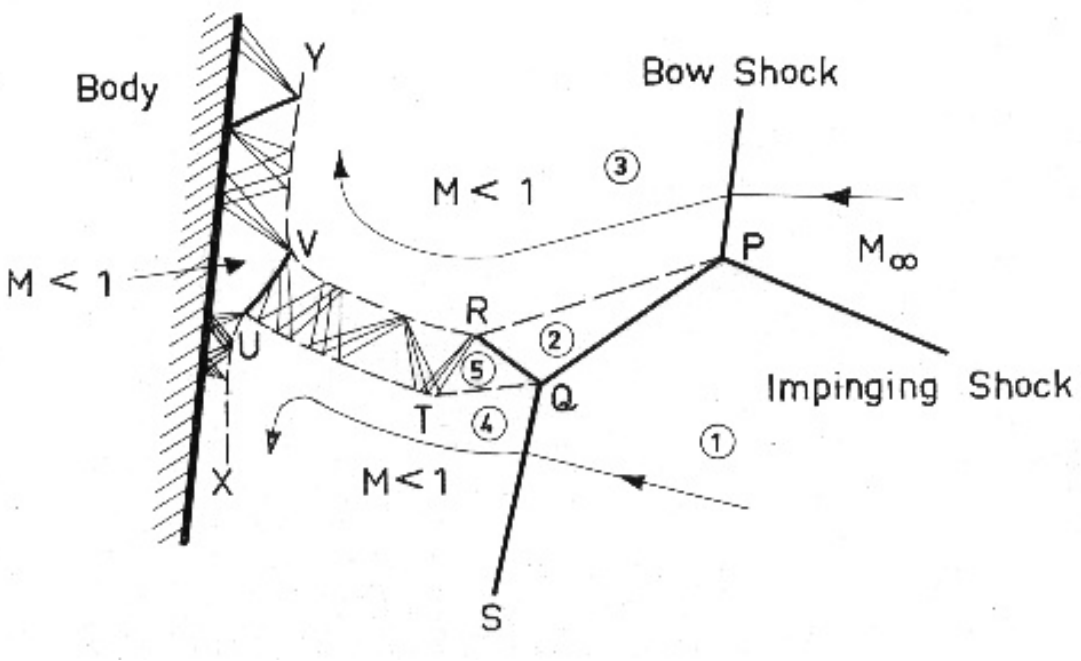
terns. These models clearly illustrated that real gas effects (in particular the variation of γ, the ratio of specific heats) play an extremely important role in determining the peak pressure loads and heat flux in an interaction region [80]. This importance of real gas effects is one of the factors that complicates the analysis of SSI in the hypersonic flow regime.

5.6.6.1 Type IV Interaction

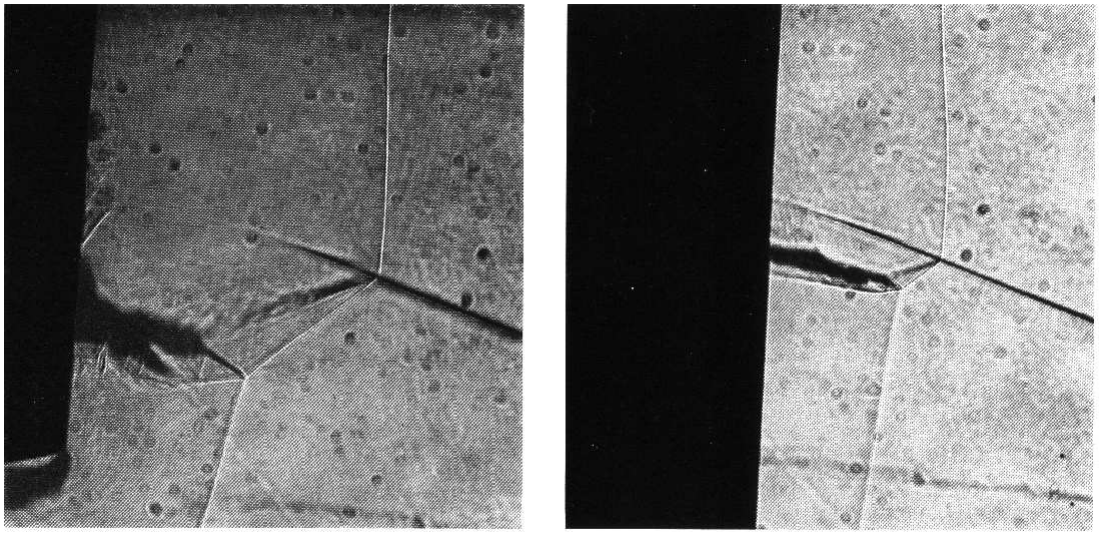
The type IV interaction pattern is shown in Figure 5.55. This interaction pattern occurs when the impinging shock intersects the body bow shock inside (its undisturbed) sonic lines. Generally a type IV interaction will occur when the impinging shock interacts with a normal shock created by the body in a region where the angle between the impinging shock and the body is large (on the order of 90°). The type IV interaction can increase local heating by over an order of magnitude. This is the most severe of the six interaction patterns and will be discussed in detail.

A characteristic feature of the type IV interaction is the strong, supersonic jet illustrated schematically by region $RTUV$. In this region a shear layer is compressed and expanded repeatedly before terminating in a normal shock (line UV) that can lie *extremely* close to the body surface. What is interesting about this interaction is the series of oblique shocks, compressions, and expansions that processes the fluid in the shear layer will in general result in a significantly lower stagnation pressure loss than occurs though the other normal shocks in the interaction. Consequently fluid with a considerable amount of energy arrives at the body and is then shocked by UV, further increasing its temperature. The point is that this particular interaction pattern allows for fluid with *much* more energy to arrive at the surface than an undisturbed normal shock would permit.

The analysis of this impingement process is complicated by another factor: the exact width of the impinging layer depends strongly on the shock standoff distance associated with points P and Q. This is particularly important in computational approaches (CFD) in which shock wave resolution is difficult. Many CFD shock–capturing techniques (essentially-non-oscillatory, artificial diffusion, flux limiting, etc...) result in shock waves



(a) Schematic Diagram



(b) Image from Edney's Experiment

Figure 5.55: Type IV shock-shock interaction [80]

that are smeared over several grid points, but have the correct behavior in the control–volume sense that the average changes across the smeared wave are correct [28]. In the particular case of a type IV interaction smeared shock waves will result in substantial local inaccuracy and uncertainty in predicting the size and strength of the impingement region. For computational approaches to be successful with these types of problems they must employ adaptive mesh refinement methods or other local solution enrichment techniques that can accurately resolve the shock waves.

Another difficulty that Edney points out in the analysis of the type IV interaction is the presence of real gas effects in the hypersonic flight regime. It is known from perfect gas compressible flow theory that the maximum attainable density ratio across a normal shock wave is 6. However, as real gas effects become important this theoretical maximum disappears and density ratios significantly higher than 6 are realizable. This has the consequence of decreasing shock standoff distance since the gas behind a normal shock can be compressed more than in the perfect gas case. This has the important effect in the type IV interaction pattern of allowing the normal shock UV to move closer to the body, which may further compress the boundary layer and increase the heating rate. Also, in the case of nonequilibrium flow, it is fairly obvious that a strongly reacting gas will be in contact with the vehicle surface after it is processed by the normal shock UV. This further complicates the accurate prediction type IV interaction patterns by both experimental and computational methods in the hypersonic regime.

5.6.6.2 Computational Simulation of a Type IV Shock-Shock Interaction

An experimental test program was conducted in 1998 by France's Office National d'Etudes et de Recherches Aérospatiales (ONERA) to investigate shock-shock interactions produced by an oblique shock impinging on the bow shock of a cylinder [81]. The test article is shown schematically in Figure 5.56. The experiments were conducted in the ONERA R5Ch hypersonic wind tunnel, which features a contoured nozzle that provides a test core 0.2 meters in diameter. The nominal stagnation conditions for the test series

were $P_0 = 2.5\,\text{bars}$ and $T_0 = 1050\,\text{K}$. Table 5.8 summarizes the relevant freestream and model parameters. Since the experiment was conducted at low density on a small model, the Reynolds number based on cylinder diameter, Re_D, is only 2,658. Consequently, the interaction region is laminar and thus amenable to computational simulation without a turbulence model. Additionally, the circular cylinder has an aspect ratio (L/D) of 6.25, so the test measurements at the center of the cylinder should be in an essentially two-dimensional flowfield [82].

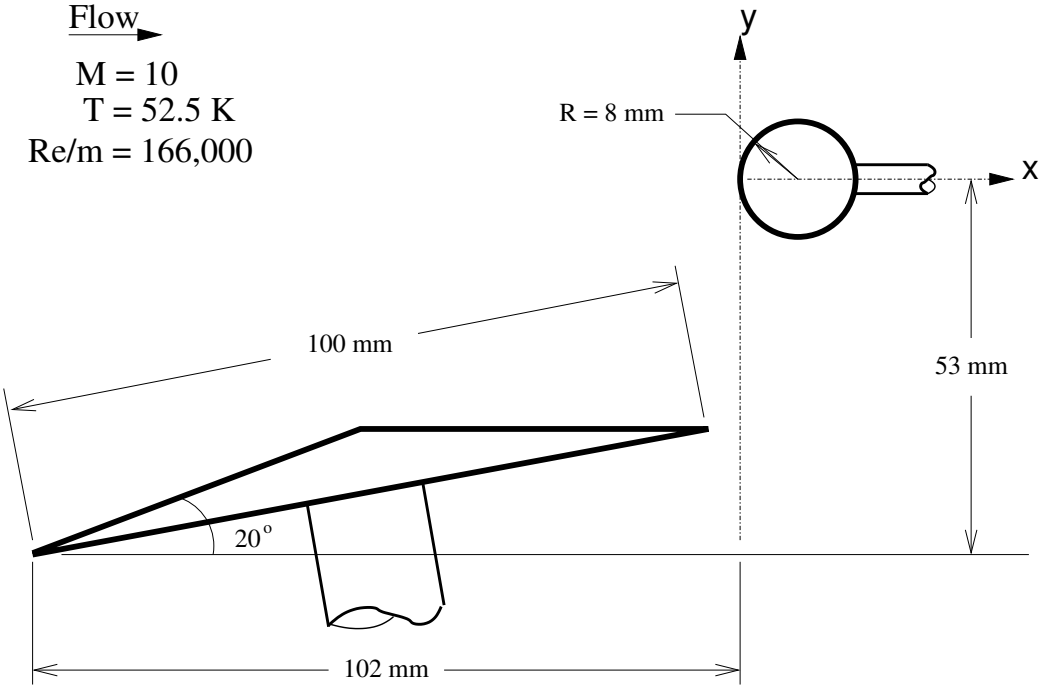


Figure 5.56: Illustration of geometry and boundary conditions for hypersonic cylinder shock/shock interaction problem

The computational domain for this case consists of the shock generator and the fore-body of the cylinder. All surfaces are treated as isothermal no-slip walls with an imposed temperature $T_w = 300\,\text{K}$. This is assumed to represent the model surface temperature at the beginning of the test run. The initial conditions are chosen to be the freestream conditions and the solution is marched in time until steady-state.

Table 5.8: Freestream parameters for hypersonic shock-shock interaction for a circular cylinder [82].

M_∞	Re_D	T_∞	T_w
10	2,658	52.5 K	300 K

Figure 5.57 shows the resulting global static temperature field for this problem. The thick boundary layers which result on both the shock generator and cylinder away from the interaction region at this low Reynolds number are evident. It is clearly necessary to include the full test geometry in this simulation due to the large displacement thickness which results on the ramp surface and the expansion downstream of the ramp corner. If either of these features were not accurately captured the location of the computed interaction region would be in error.

Figure 5.58 details the local features which occur in the interaction region. The oblique incoming shock and large-scale bow shock displacement are clearly evident. A severe temperature gradient at the cylinder surface is apparent, which is expected to result in high, localized heat transfer rates. Figure 5.59 compares the predicted and measured surface pressure and heat transfer, respectively. The values are scaled by the corresponding undisturbed cylinder stagnation point values.

Figure 5.60 presents horizontal cuts through the shock layer and comparisons with measured temperature and density at several stations in the interaction region. The measurements are made with the dual line coherent anti-Stokes scattering (DL-CARS) technique [81, 82] which can be used to provide instantaneous measurements of temperature and pressure in low pressure supersonic flowfields. Several pulsed lasers of different frequencies are used to probe the flow volume of interest at a discrete instant in time. For steady applications the signal from several pulses can be averaged to improve the signal to noise ratio of the measurement. The probed volume is 40 mm in the spanwise direction by approximately 0.2 mm diameter, hence the data are spatially averaged over approximately

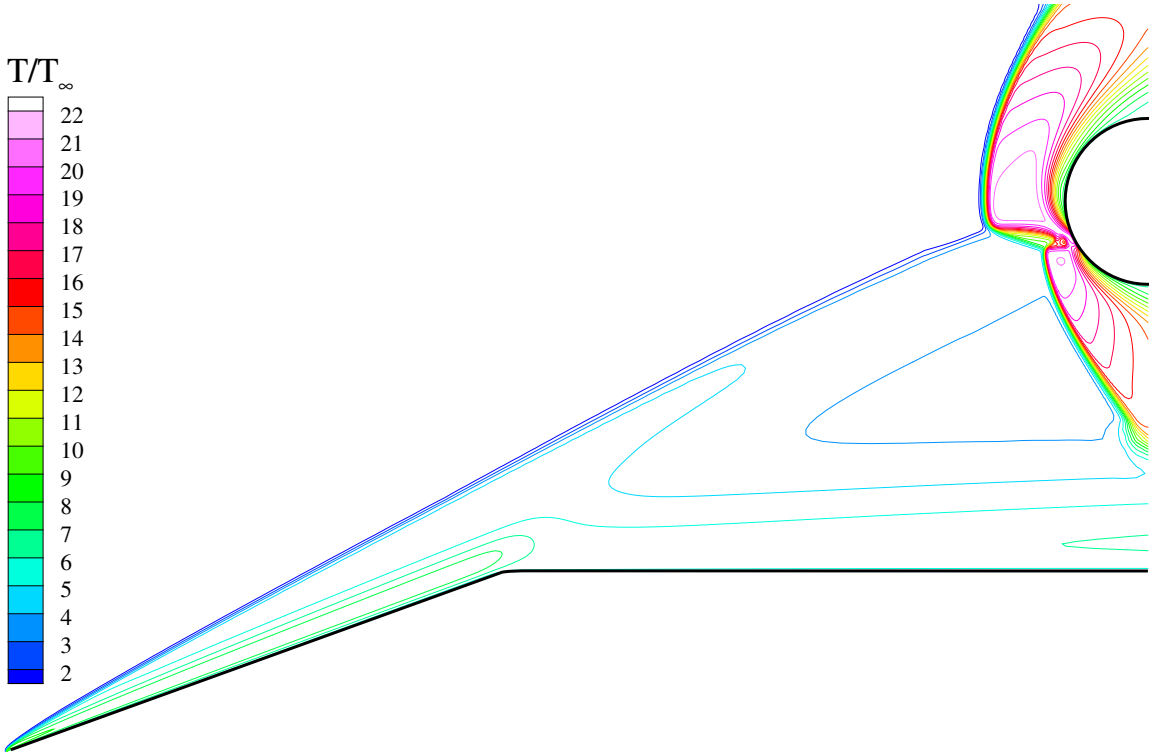
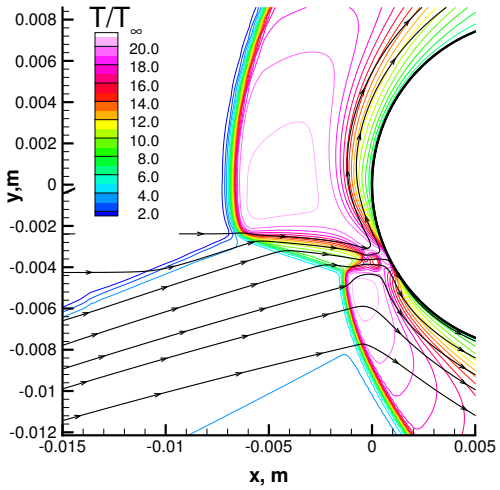
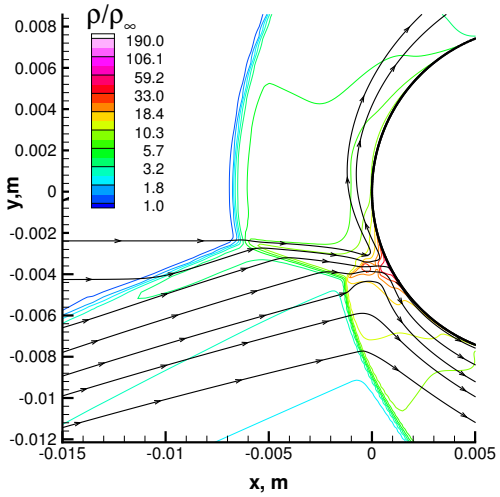


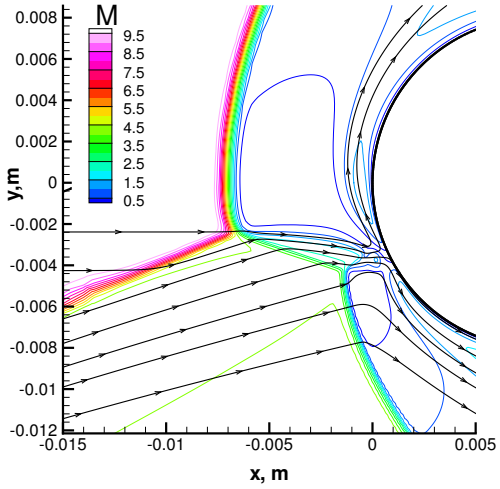
Figure 5.57: Global flowfield for hypersonic cylinder shock/shock interaction problem



(a) Temperature Contours



(b) Density Contours



(c) Mach Contours

Figure 5.58: Illustration of flowfield in interaction region for hypersonic cylinder shock/shock interaction problem

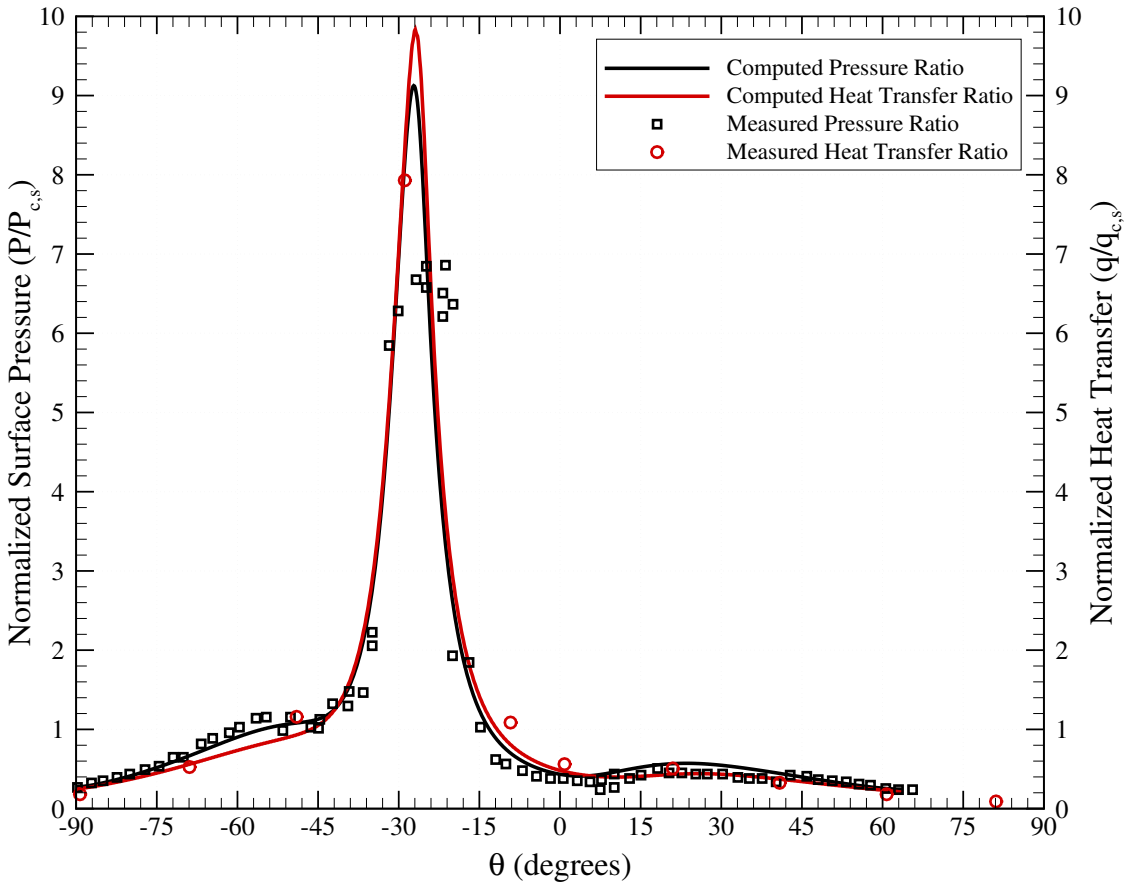


Figure 5.59: Comparison of predicted and measured surface pressure and heat transfer ratios. Values are normalized by corresponding undisturbed cylinder stagnation point values.

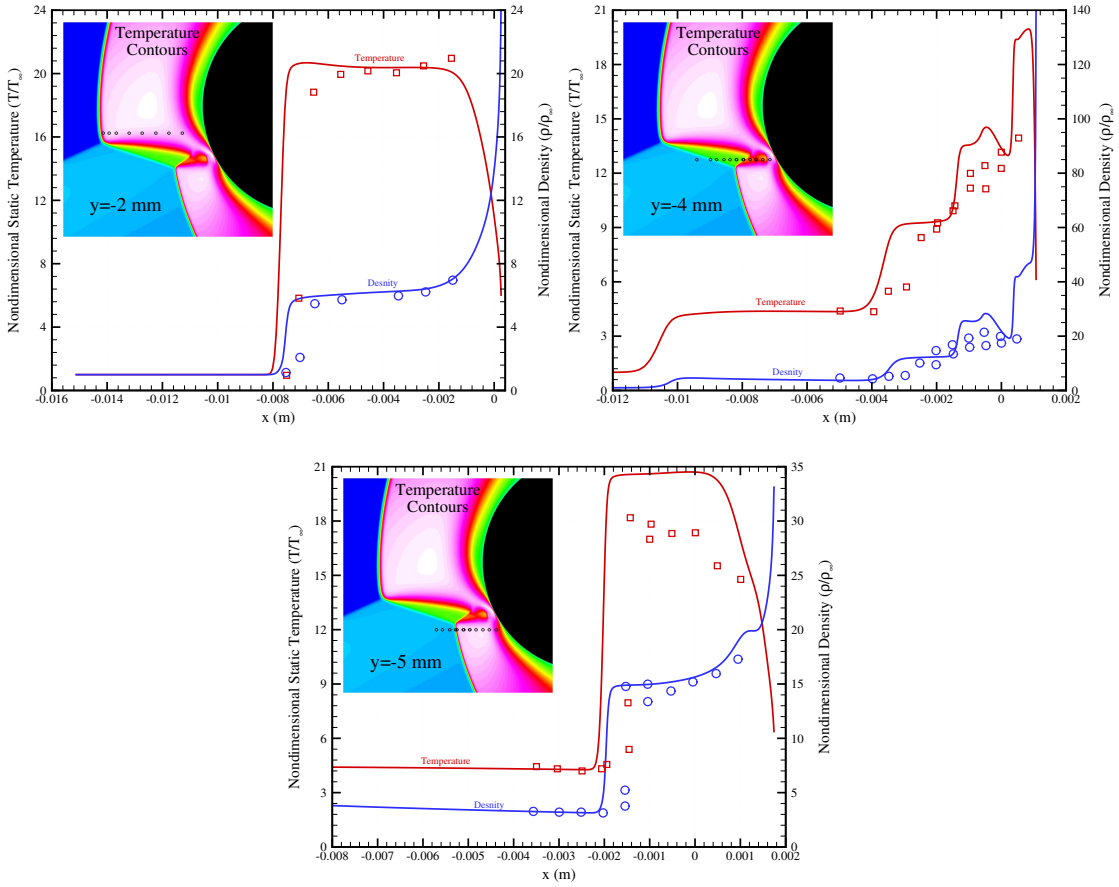


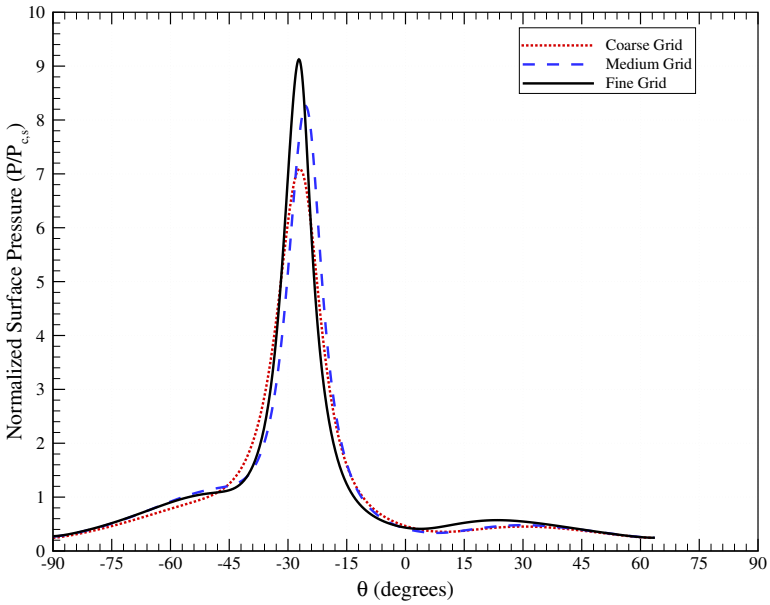
Figure 5.60: Temperature and density profiles in the type IV shock interaction region.

0.2 mm [81]. The symbols inset in the figures depict the location and approximate spatial resolution of the probed regions.

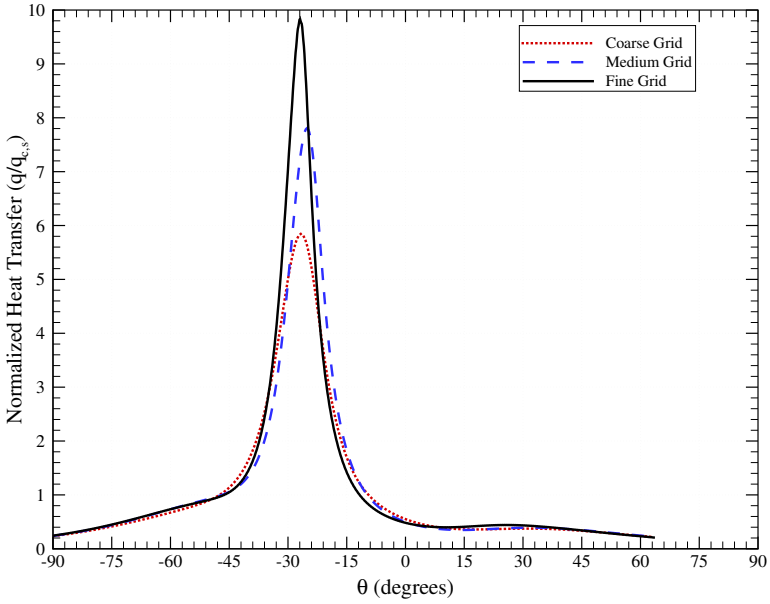
Figure 5.61 presents mesh convergence studies for the surface properties of interest. In Figure 5.61(a) the surface pressure distribution is shown for a sequence of three meshes. Similarly, the heat transfer distribution is shown for the same sequence of meshes in Figure 5.61(b). The coarse mesh for this case consists of 44,815 elements and 215,380 degrees of freedom. The medium and fine meshes correspond to one and two levels of uniform refinement for a total of 856,740 and 3,417,415 degrees of freedom, respectively. It is interesting that in both cases the convergence of the quantities of interest is from below. One possible explanation for this behavior is that as the mesh is refined the inherent diffusion in the shock-capturing scheme is reduced, thus allowing higher enthalpy flow to arrive at the cylinder surface in the interaction region. Interestingly, the general location of the interaction is well predicted by all three grid levels, but the magnitude of both the pressure and heat transfer augmentation is underpredicted on the coarse meshes. This is particularly troubling for designs which must contend with a type IV shock interaction because the computational requirements for obtaining mesh-converged solutions is substantial.

5.6.6.3 Type VI Interaction

The type VI interaction pattern is shown in Figure 5.62. This interaction pattern results when the impinging shock meets the bow shock well above its upper sonic line. A characteristic feature of this interaction pattern is the expansion fan between regions 2 and 4. Additionally, there is a shear layer that results from this interaction pattern. Both the expansion region and shear layer may impact the body downstream of the interaction region. However, it is notable that these features have not been observed to appreciably affect the surface heating [80]. The case of a three-dimensional type VI shock interaction will be considered in the following section in the context of the Space Shuttle Orbiter at reentry attitude.

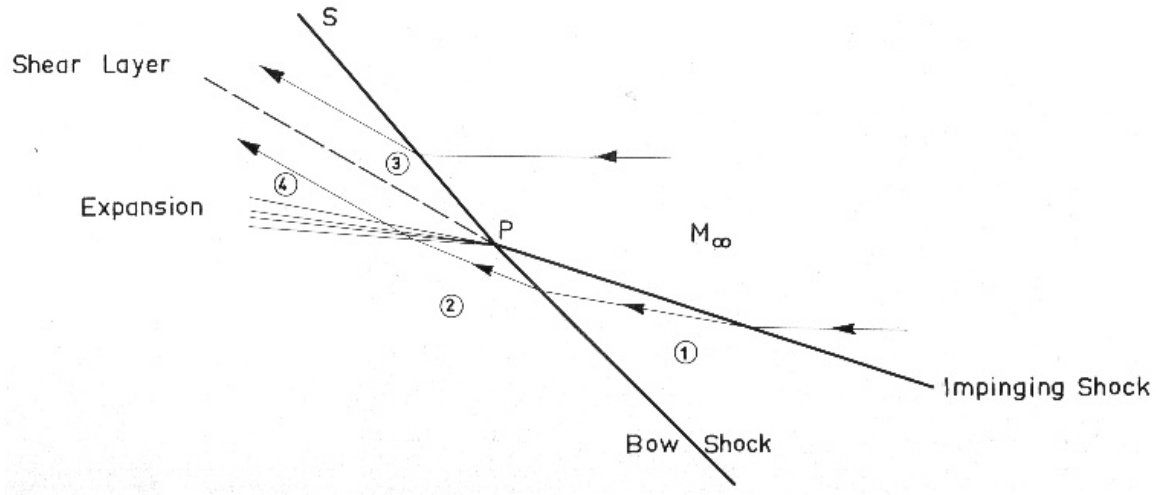


(a) Pressure

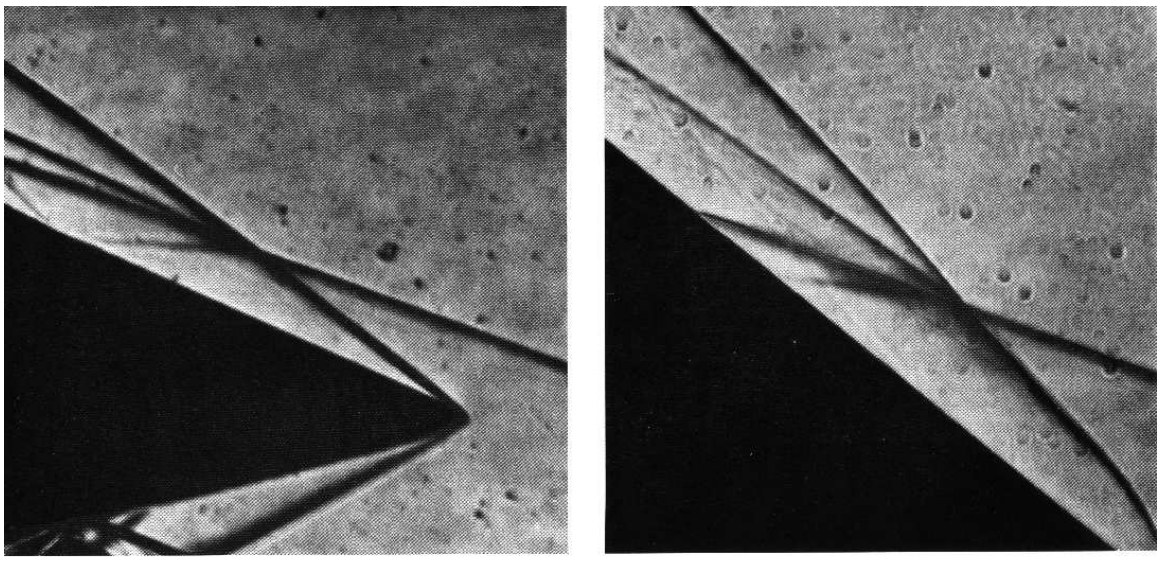


(b) Heat Transfer

Figure 5.61: Mesh convergence for surface quantities.



(a) Schematic Diagram



(b) Image from Edney's Experiment

Figure 5.62: Type VI shock-shock interaction [80]

5.6.7 Space Shuttle Orbiter

Reentry conditions pose a particular challenge to computational simulation. During the process of reentry a spacecraft progresses through all phases of atmospheric flight, from free molecular flows, through hypersonic flight, to low speed transonic and ultimately incompressible flows. While this is clearly a transient process, in current practice the analysis is typically performed at a series of static conditions. That is, the reentry profile is modeled as a series of quasi-steady states. The high-Mach hypersonic regime is of interest in the current study as it generally provides the design conditions for the vehicle thermal protection system.

Reentry vehicle design has progressed over the past 50 years from simple asymmetric designs to complex, winged vehicles. Consider for example the evolution of the U.S. manned spaceflight programs: The Mercury program used simple asymmetric capsules which reenter on ballistic trajectories. The Gemini and Apollo programs used capsules with an offset center of gravity, which provided cross range capabilities. The Space Shuttle Orbiter and follow-on designs employ winged lifting bodies with increased cross range capabilities. This design progression was driven primarily by end-of-mission landing accuracy requirements. The cross range design requirements for the Orbiter were driven by the desire to launch into a descending-node polar trajectory from Vandenberg Air Force Base in California, and returning for landing at the same location after one orbit. While this capability has never been used, this cross range capability is still desirable as it translates into less restrictive reentry windows for landing on a fixed runway.

The geometric complexity of the winged reentry vehicle design allows for shock interactions which must be adequately understood and properly accounted for during vehicle design. In the case of the Orbiter, the peak heat flux to the vehicle occurs on the wing leading edge as a result of a type VI bow shock/wing shock interaction. The focus of the current simulations is to demonstrate the applicability of the finite element discretization

developed in this work to this class of problems. To this end, the Orbiter at wind tunnel conditions which simulate reentry is simulated at a range of angles of attack. The freestream conditions for this case are listed in Table 5.9.

Table 5.9: Freestream parameters for wind tunnel simulation of Space Shuttle Orbiter reentry.

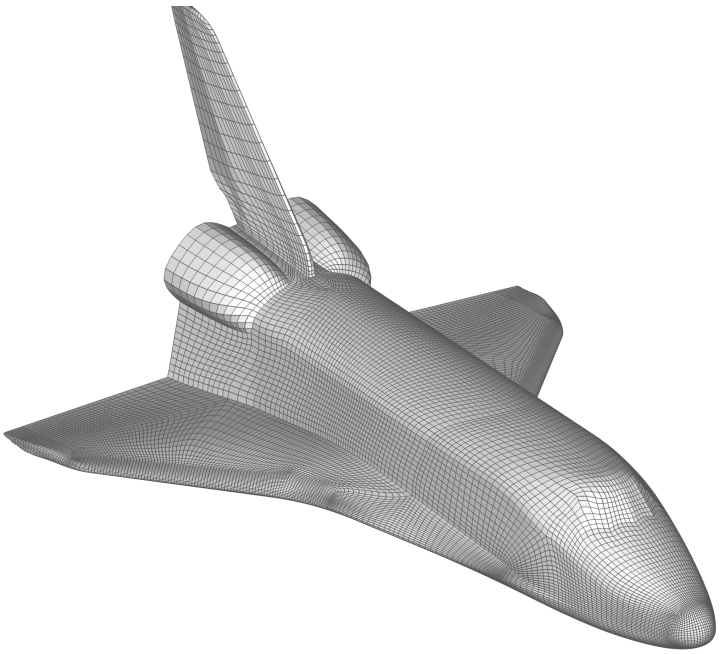
M_∞	Re_L	T_∞	T_w	α
10	4.4×10^5	57.8 K	300 K	40°

5.6.7.1 Computational Mesh

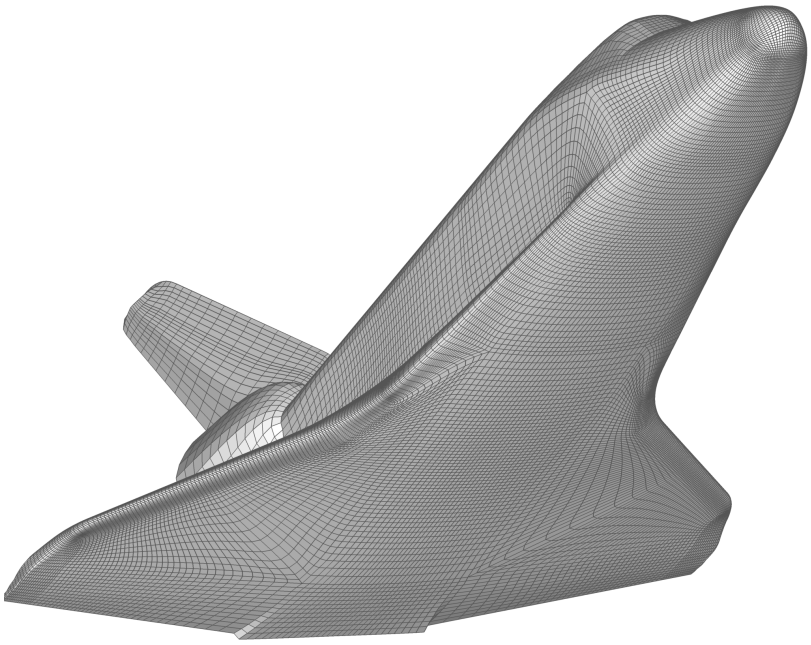
The computational mesh used in this simulation is derived from a multi–block structured topology. The mesh contains $1,047,360$ elements and $1,079,910$ nodes for a total of $5,399,550$ degrees of freedom in the simulation. The mesh was partitioned and run on 128 processors of the *lonestar* supercomputer located at The University of Texas at Austin's Texas Advanced Computing Center.

The surface mesh for this case is shown in Figure 5.63. The volume mesh contains 64 cells in the off-body direction, and the outer boundary is tailored to contain the forebody bow shock and afterbody expansion region for a range of Mach numbers at reentry attitude (characterized by an angle of attack, α, of approximately 40°).

The mesh used here is adapted from Return-to-Flight common grid, which incorporated lessons learned from the Columbia Accident Investigation common grid [83]. As the names imply, these grids were developed with the intent of supporting a range of block-structured flow solvers. It is truncated downstream of the body flap hinge line and includes no downstream wake region to reduce the overall problem size. This modeling assumption is made because these features do not affect the wing leading edge of the vehicle, which is the region of the interest in this case. Also, since the Orbiter flies with negligible



(a) Upper surface



(b) Lower surface

Figure 5.63: Space Shuttle Orbiter surface mesh.

sideslip during the peak heating phase of reentry, y-symmetry is assumed and only half of the vehicle is modeled.

It should be noted that omitting the wake of the vehicle induces a modeling inconsistency in that there is no pressure coupling at the trailing edge of the wing. In reality there is pressure coupling at the trailing edge of the Orbiter elevon. This coupling can induce substantial changes in the separation lines on the aft portions of the upper wing surface but has no effect on the wing leading edge.

5.6.7.2 Results

Results from this simulation are shown in Figures 5.64, 5.67, and 5.68. Figure 5.64 depicts the static surface pressure and several streamlines in the flowfield. The pressure contours are spaced logarithmically to detail the upper surface features. The streamlines are colored by flowfield static temperature. Two vortex pairs develop as the flow expands toward the leeside of the vehicle. One pair results from flow expanding downstream of the crew cabin over the payload bay doors. This flow structure is similar to the vortex structure which results for slender bodies (such as missiles) at angle of attack. Note that this vortex pair ultimately interacts with the vertical stabilizer and is processed by a shock wave emanating from the OMS pods, resulting in visibly increased static temperature. The flow expansion downstream of the crew cabin is supported through anecdotal evidence by observations of ice on the payload bay doors after landing, despite reentry temperatures approaching 3,000 °F on the lower surface.

The other vortex pair is a result of the double-delta wing design. Flow is expanded around the "chine" by the favorable windward-to-leeward pressure gradient. This expansion, and the sweep of the chine, induces the rotation which ultimately produces these vortices. It is interesting to note the elevated pressure on the side fuselage, which is essentially parallel to the freestream velocity. The chine vortex "scrubs" the sidewall, causing this elevated pressure as well as additional heating. The pressure is reduced towards the aft half of the fuselage because the wing is directing flow away from the vehicle. This pres-

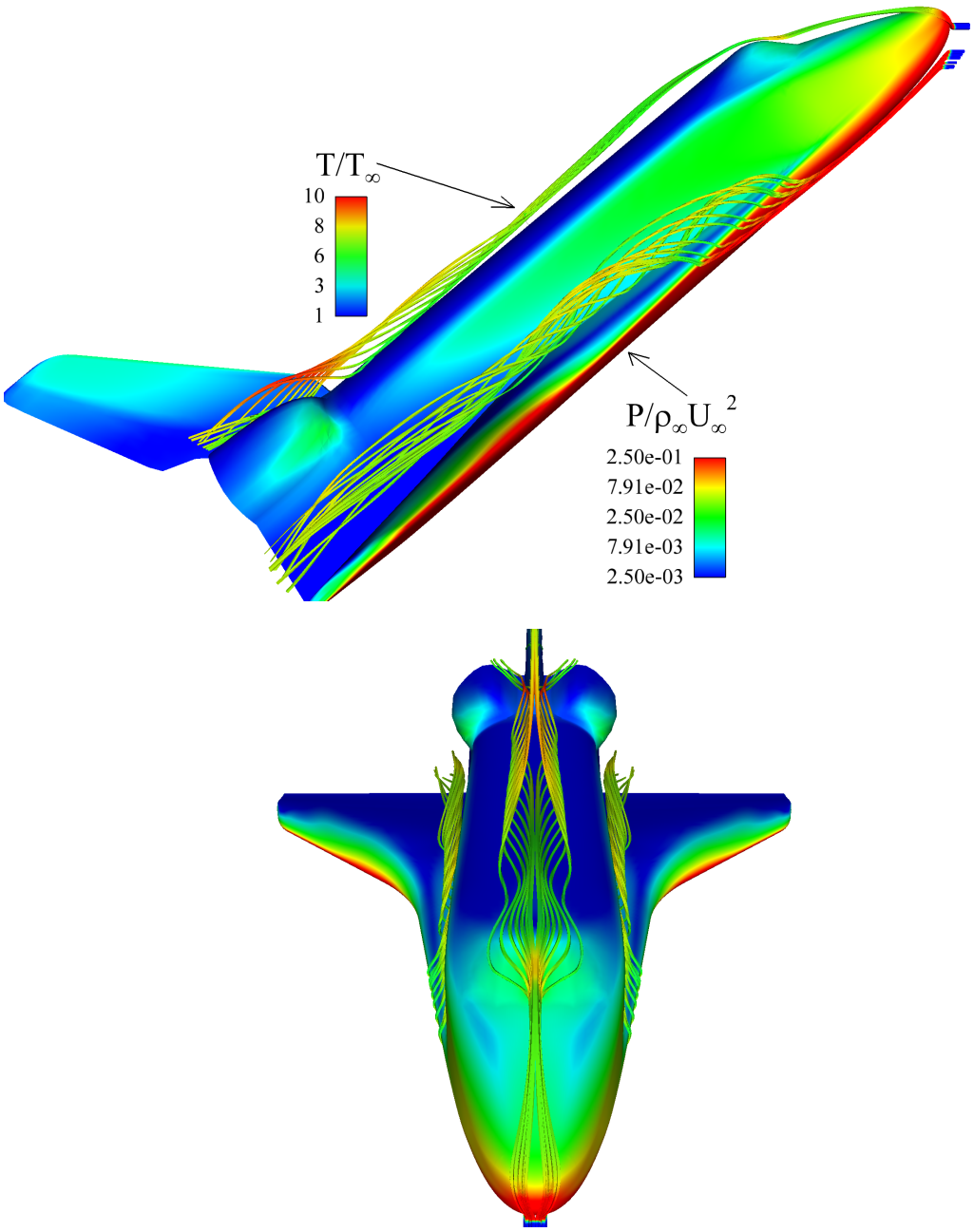


Figure 5.64: Space Shuttle Orbiter wind tunnel simulation. Streamlines are colored by nondimensional static temperature. Surface contours are of nondimensional static pressure and are spaced logarithmically to illustrate upper surface flow features.

sure distribution influences the heat transfer on the vehicle as well. The favorable pressure gradient on the side fuselage allows for an attached boundary-layer to develop.

Figure 5.65 shows two Orbiters, *Columbia* and *Discovery*, immediately after reentry. The side fuselage of the vehicles are clearly discolored in the region of attached flow. This discoloration is due primarily to the outgassing of surface adhesive material which is used to bond the thermal protection system to the vehicle structure. These gaseous products are convected downstream and yield a qualitative indication of the local flow structure. The similarity between the pressure distribution predicted in Figure 5.64 and the discoloration observed in Figure 5.65 helps provide confidence in the extent of the separated afterbody flow predicted by the computation.

Finally, elevated surface pressure occurs on the portion of the OMS pod that is included in the analysis. This is not surprising, since the OMS pods induce a normal shock at the forward outboard corner. The initial thermal protection system design for the Orbiter used white tiles on this portion of the vehicle. These white tiles are visible in Figure 5.65(a), which shows the landing of STS-1, the first Shuttle flight. The surrounding portions of the vehicle are covered with insulating blankets due to the relatively benign heating environment. As a weight-savings measure these tiles were replaced with blankets on the Space Shuttle Orbiter *Challenger*. However, the blanket design proved inadequate and was replaced after *Challenger's* maiden flight with a small number of black tiles similar to those used on the lower surface, as shown in Figure 5.65(b). These black tiles have a higher surface emissivity and thus release heat via radiation more effectively.

According to the NASA STS-6 mission report, "the AFRSI (advanced flexible reusable surface insulation) on the OMS pods experienced severe damage on the forward portion and minor damage at other locations." This damage was classified as STS-6 in-flight anomaly number STS-6-V-05 [85]. Figure 5.66 shows both the starboard and port OMS pods after the STS-6 landing. The composite substructure is clearly visible in the figures, indicating that the AFRSI protective blanket has been completely removed.

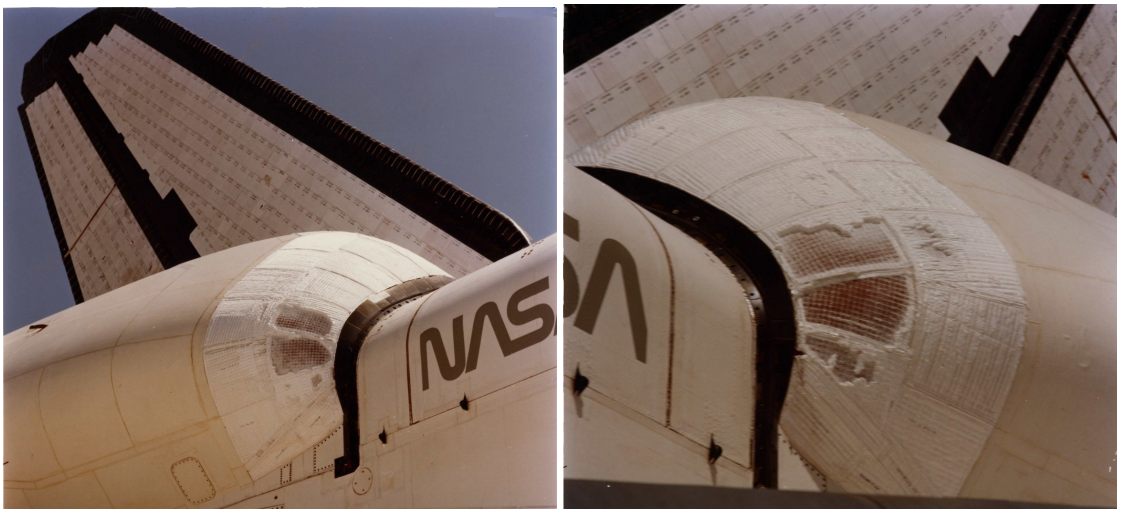
(a) STS-1. The Space Shuttle Orbiter *Columbia* touching down at Edwards Air Force Base, April 14, 1981.

(b) STS-114. The Space Shuttle Orbiter *Discovery* is towed from the runway at NASA's Dryden Flight Research Center, August 9, 2005.

Figure 5.65: Space Shuttles *Columbia* and *Discovery* at landing. Note the side fuselage tile layout and discoloration. Also of interest is the difference in OMS pod TPS design [84].

(a) Starboard OMS pod

(b) Port OMS pod

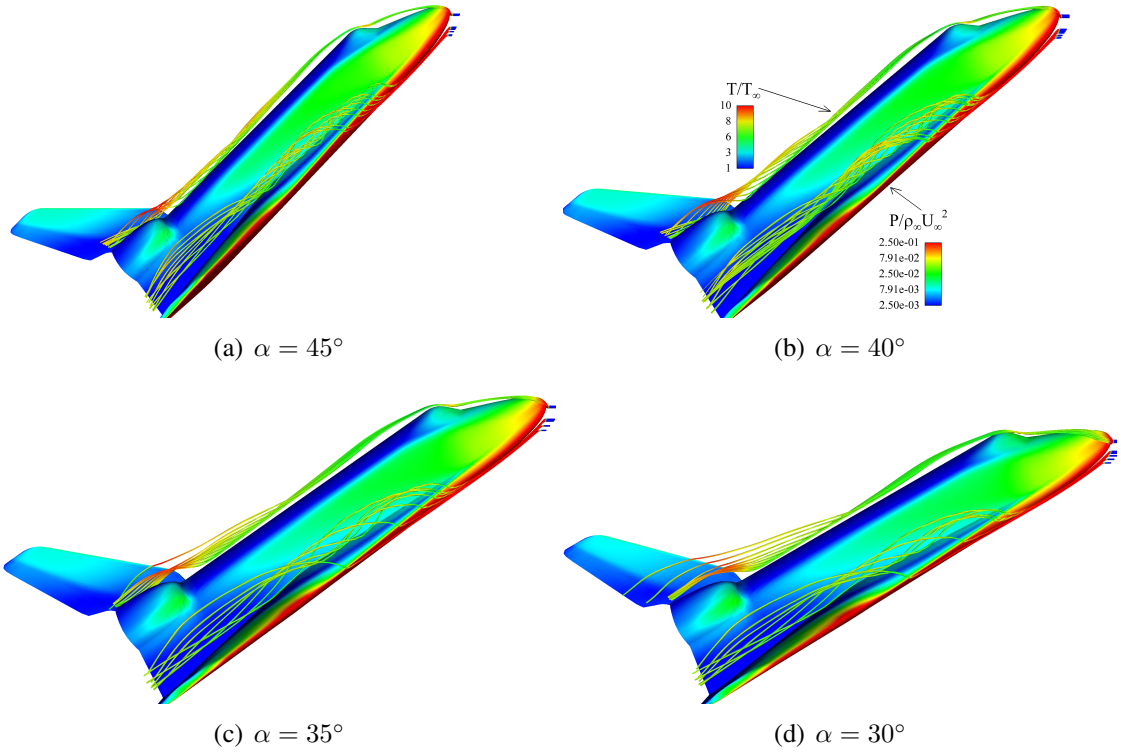


(c) Starboard OMS pod closeup

(d) Port OMS pod closeup

Figure 5.66: *Challenger* on the tarmac at Edward's Air Force Base after STS-6. Note the local thermal protection system damage on the outboard forward section of the OMS pods.

Figure 5.67 shows a series of solutions at $\alpha = 30^\circ - 45^\circ$. While not relevant to nominal reentry, these lower angles of attack are within the abort flight envelope capabilities of the Orbiter. For example, during a trans-Atlantic launch abort the Orbiter may fly at these lower angles of attack in order to reach landing sites in Spain or northern Africa. At the lower angles of attack the pressure distribution on the OMS pod is clearly higher than at the higher angles of attack. This increased pressure results because at the lower angles of attack the flow impinging on the OMS pod maintains more of the freestream momentum than at the higher angles of attack.



(a) $\alpha = 45^\circ$ (b) $\alpha = 40^\circ$

(c) $\alpha = 35^\circ$ (d) $\alpha = 30^\circ$

Figure 5.67: Angle of attack sweep for the Space Shuttle Orbiter.

The intensity of both the payload bay and chine vortices increases with angle of attack. As the angle of attack increases the windward-to-leeward pressure gradient increases, resulting in increased rotation as the flow expands around the crew cabin and chine region.

The increased rotation is evident from the streamlines in the Figure.

Figure 5.68 shows planar cuts through the bow shock/wing shock interaction region. The contours on the vehicle surface correspond to static pressure, which clearly peaks at the nosecap and in the shock interaction region. The static pressure in the flowfield depicts the elevated pressure in the interaction region. This augmented pressure thins the boundary layer and causes increased heating to the wing leading edge. The wing leading edge sweep angle is $45°$, and its chordwise curvature causes it to respond conceptually as a swept cylinder. This analogy suggests that the post-shock flow would therefore be supersonic, which is clearly evident from the Mach distribution shown in the Figure. Note the similarity between the Orbiter bow shock/wing shock interaction and the type VI experimental schlieren image obtained by Edney which is shown in Figure 5.62(b) for the case of a swept cylinder.

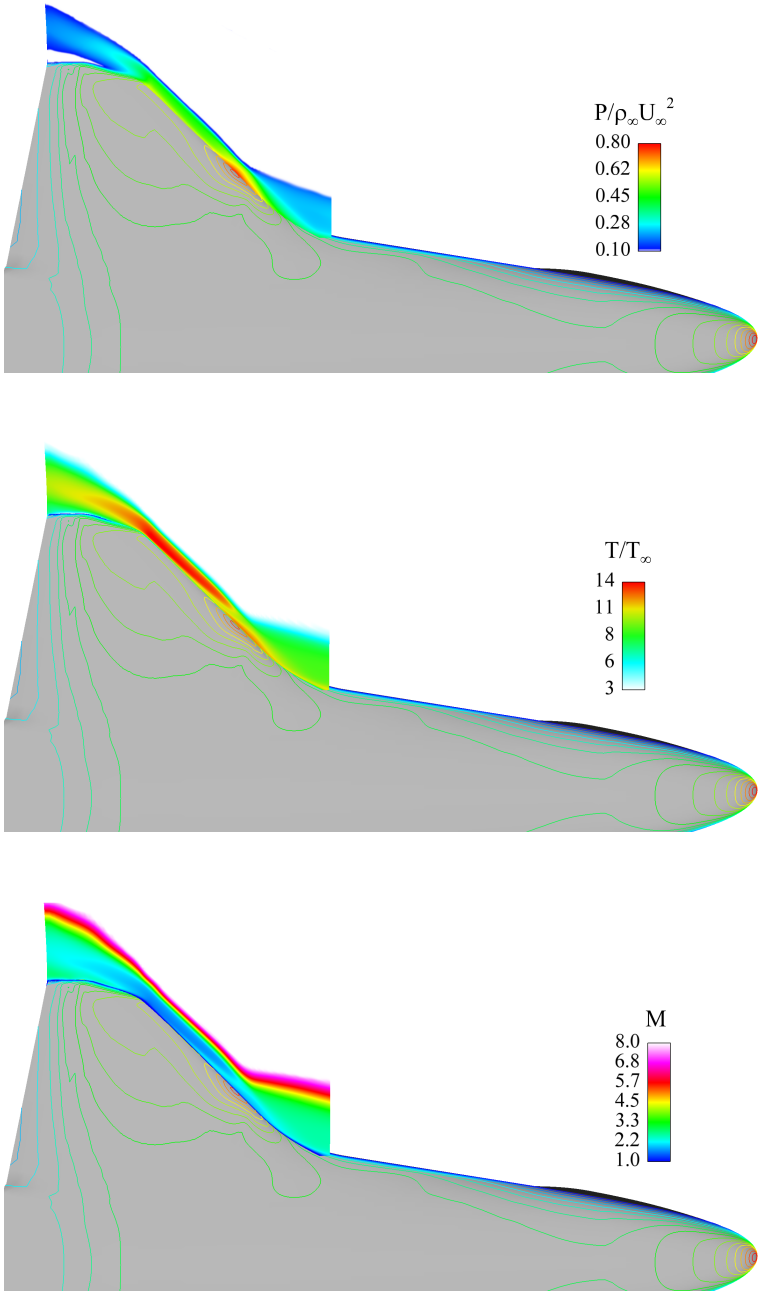


Figure 5.68: Space Shuttle Orbiter wind tunnel simulation. Planform view detailing shock-shock interaction region. The vehicle surface displays pressure contours.

5.6.8 Orbital Space Plane Design Concept

Prior to the Columbia tragedy, NASA was engaged in a design study to develop the "Orbital Space Plane." This vehicle was not intended to have the payload capacity of the Space Shuttle System, but rather its primary purpose would be to carry crew to low Earth orbit. One design concept is shown schematically in Figure 5.69. The vehicle is a classic lifting body, employing a single delta–wing configuration. This eliminates the shock–shock interaction inherit in the double delta–wing Orbiter design. The vehicle was designed to trim at a 40° angle of attack through the hypersonic regime. The other parameters of interest, relevant to the post-peak heating portion of reentry, are listed in Table 5.10.

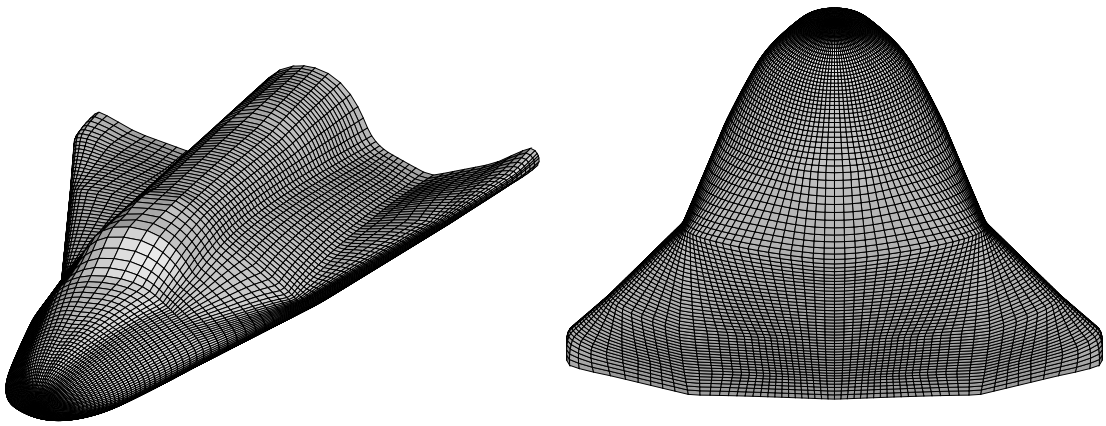
Table 5.10: Freestream parameters for orbital space plane reentry.

M_∞	Re_L	T_∞	T_w	α
6	5×10^5	200 K	500 K	40°

5.6.8.1 Computational Mesh

The computational mesh used in this simulation is adapted from a single–block structured topology. The surface mesh is shown in Figure 5.69. The grid contains a singular axis at the nose of the vehicle, which is typically problematic for block-structured solvers due to the non-invertability of the mapping from computational to physical space. In the current approach prismatic elements are placed around the axis, which is then no longer a singularity, and no numerical difficulties are encountered. The mesh contains $253,704$ hexahedral and prismatic elements with $263,700$ nodes yielding a total of $1,318,500$ degrees of freedom in the simulation. The mesh was partitioned and solved on 44 processors of a parallel cluster.

One interesting observation is that while the prismatic approach removes the singular axis present in the structured grid mapping, it does so by introducing a large number of



(a) Upper surface (b) Lower surface

Figure 5.69: Orbital space plane surface mesh.

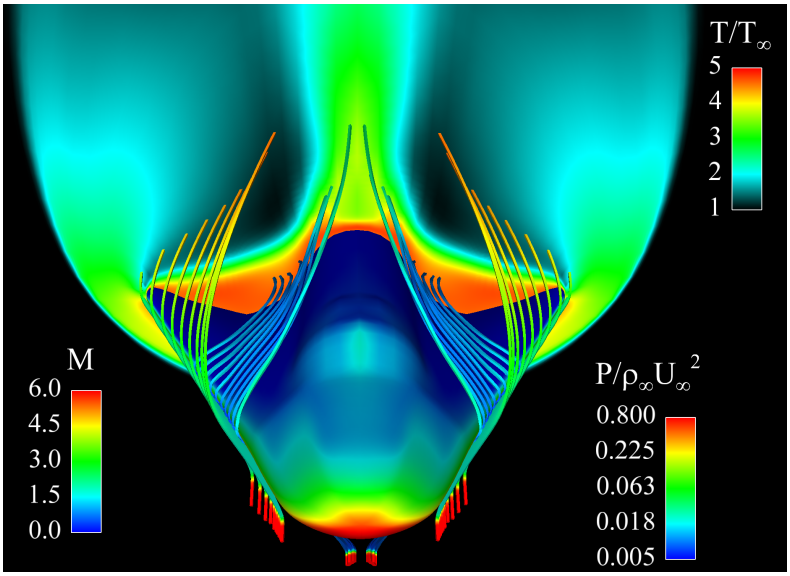
coupled elements. As mentioned previously, a standard piecewise linear Lagrange basis is used to discretize (5.50). The support of a given Lagrange finite element basis function extends over the patch defined by all elements connected to a given node. As a consequence, the large number of triangles which connect at the axis are all coupled together. This results in a particularly large row in the resulting sparse matrix which is constructed for the discrete system. While not prohibitive in this case, for a finer grid this approach could lead to a memory imbalance between processors as the large row will be wholly owned by one processor. A simple approach for addressing this situation will be considered in Section 5.6.9.

5.6.8.2 Results

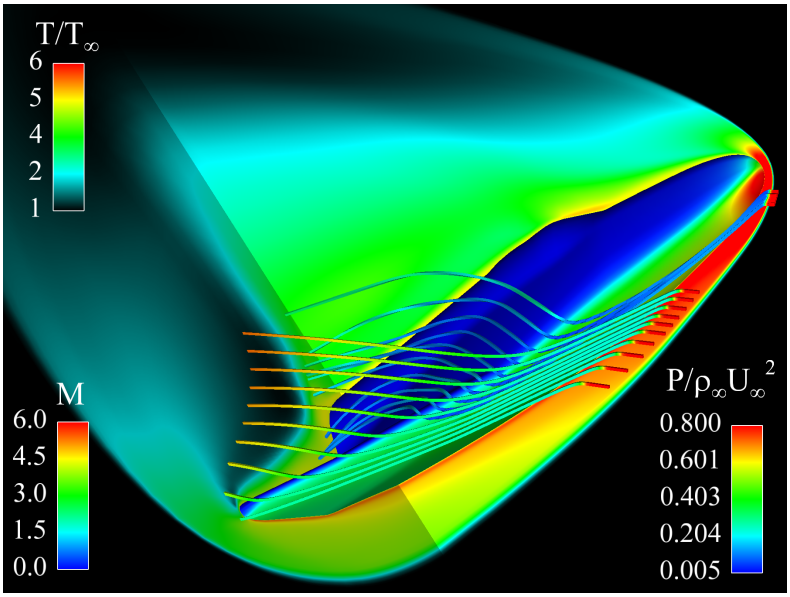
The flowfield static temperature and surface pressure are shown in Figure 5.70. The streamlines in the figure are colored by Mach number. The flowfield temperature illustrates the deep expansion on the leeside of the vehicle. The flow expands rapidly from the windward to leeward surface. This expanded flow is then processed by an oblique

shock as it approaches the leeside pitch plane, resulting in increased temperature in this region. The fairly constant sweep, wide delta wing configuration of this vehicle minimizes the vorticity affecting the upper surface at reentry attitude. The streamlines plotted in the figure indicate that, compared to the previous case of the Orbiter, there is relatively little rotation induced as the flow spills overboard into the wake region.

The surface pressure is largely Newtonian, and the large windward to leeward surface pressure discrepancy is clear. This pressure distribution is what induces the lift and drag on the vehicle, causing the deceleration force required for a reentry vehicle. The results in Figure 5.70(a) show a slight pressure increase on the upper surface caused by a weak canopy shock. This local increase in pressure would need to be included in any window design, although the design loads in this region would most likely be driven by the maximum dynamic pressure condition at launch, or later in the reentry trajectory in the transonic regime.



(a) Upper Surface



(b) Pitch Plane

Figure 5.70: Flowfield temperature and surface pressure for orbital space plane configuration. Streamlines are colored by Mach number.

5.6.9 X-38 Crew Return Vehicle

During the development of the International Space Station (ISS) it was recognized that there must be some means for quickly returning crew members to Earth in the case of medical or mechanical emergency. Fulfilling this requirement means that a spacecraft capable of reentry must be docked to ISS at all times. This vehicle must be capable of supporting the full station crew and safely returning them to Earth. Further, this "lifeboat" vehicle must be capable of remaining on-orbit for extended periods of time and still be capable of fulfilling this contingency crew support and reentry capability.

The Russian Soyuz capsule was envisioned as filling this role during the early years of station operations. The Soyuz can only support a crew of three, hence during these early years the maximum crew size of any station expedition would be limited to three. This was not seen as a constraint during the early assembly phase of the ISS, as there would not be sufficient on-orbit resources to support a larger crew. As the station became operational, however, a target crew size of seven was desired. Clearly, a new crew return vehicle beyond the Soyuz would be required to support this larger crew size. NASA's Lyndon B. Johnson Space Center initiated the development of the X-38 Crew Return Vehicle (CRV) in 1995 to meet this need.

The X-38 CRV was designed as a lifting body which would be capable of returning a crew of seven to land within hours of undocking from station. In order to support immediate return in case of crew illness or injury, it was desired that the CRV be capable of anywhere-landing, much like the Soyuz. Further, to minimize crew recovery time, some appreciable lift-to-drag ratio (L/D) was sought so that the vehicle could target optimal landing sites. The X-38 CRV was modeled after early work on lifting bodies, which occurred in the X-24 and related programs. Precision landing was to be made possible by a steerable parafoil recovery system. The X-38 in flight and lading during two drop-tests is shown in Figure 5.71.

(a) Separating from NASA's B-52.

(b) Landing.

Figure 5.71: X-38 Crew Return Vehicle drop tests [84].

The X-38 lifting body design shares some similarities with the Space Shuttle Orbiter. During the peak heating portion of reentry, for example, the vehicle maintains fixed pitch via "body flap" deflection. Lateral-directional control is achieved via a series of roll-reversal maneuvers. Unlike the Orbiter, however, the X-38's body flap is the only active control surface during reentry. (The Orbiter has elevons and an active set of reaction control thrusters which provide roll control.) In order to maintain roll authority the X-38 used a split body flap design, which can be seen in Figure 5.72. In this design, the two halves of the body flap may be articulated separately. By deploying the two flaps at different angles, a rolling moment will result due to the higher surface pressure on the flap with higher deflection. A slight differential deflection is visible in Figure 5.72(a).

One particular challenge posed by the split body flap design is the seam that is necessarily introduced on the vehicle centerline. The extreme pressure change from the front-to-back of the deflected body flap will induce some amount of high-enthalpy flow through this gap. Therefore, the body flap and gap regions were the focus of a number of experimental test programs. Recalling the 15° compression ramp of Section 5.6.2, it is also expected that the adverse pressure gradient set up by the deployed body flap may induce boundary-layer separation. As was the case for the compression ramp, there may also be some localized augmentation in heat transfer in the reattachment region. Depending on the body flap deflection (and hence size of the separated region), shock interaction similar to that seen in Section 5.6.3 for the case of an axisymmetric cylinder-flare may also occur.

NASA eventually partnered with the European Space Agency (ESA) during the design and development phase of the program. Minor modifications were made to the baseline design such that the vehicle could be launched atop the French Ariane booster. The geometric differences visible at the aft of the vehicle in Figures 5.71(a) and 5.72(a) are due to the modifications required to mate the X-38 with the Ariane. In addition, the streamwise structure of the vehicle was strengthened in order to react launch loads.

Unfortunately, the X-38 program was canceled before hypersonic reentry flight data were obtained. Still, during its design and development phase a number of aerother-

(a) Immediately after separation.

(b) In free flight.

Figure 5.72: X-38 in flight. Note the deployed split body flap [84].

modynamic wind tunnel tests were conducted to obtain validation data for CFD models. An equally extensive aerodynamic test program was completed, which culminated in the previously-mentioned drop-tests of the combined vehicle geometry and parafoil. In this work the X-38 is modeled at wind tunnel conditions chosen to simulate reentry. The relevant freestream parameters are listed in Table 5.11.

Table 5.11: Freestream parameters for wind tunnel simulation of X-38 Crew Return Vehicle reentry.

M_∞	Re_L	T_∞	T_w	α
10	7.5×10^5	50 K	300 K	40°

5.6.9.1 Computational Mesh

The computational mesh used for this case was adapted from a single-block structured grid topology.[2] The surface discretization used in the original grid is shown in Figure 5.73. The single block topology, including axis singularity, is clearly visible. One interesting feature of this grid is the large number of points used in the circumferential direction. This grid density was required to accurately capture the fins and body flap geometry. As a consequence, however, there is substantially more mesh on the forward portion of the vehicle than required to achieve an accurate solution.

This grid was taken as a starting point for a hybrid hexahedral/prismatic mesh. The resulting mesh, shown in Figure 5.74, substantially reduced the number of points used on the smooth forebody. The original grid was halved in the circumferential direction repeatedly. The resulting disjoint structured quadrilateral grids were then "stitched" together with unstructured triangulated regions. Note that as a consequence of this hybrid mesh approach the number of triangles in the nose patch is reduced by a factor of eight. This corresponds

[2]Mr. Charles H. Campbell of the Aeroscience & Flight Mechanics Division at NASA's Lyndon B. Johnson Space Center provided the original structured grid.

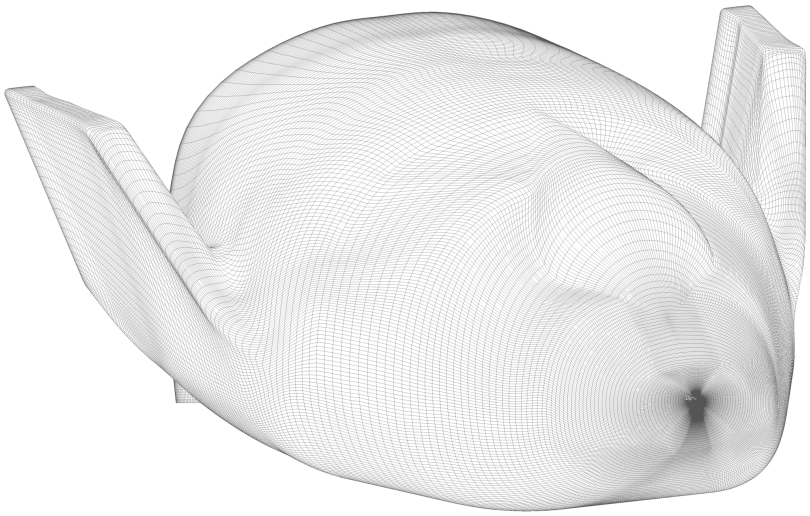
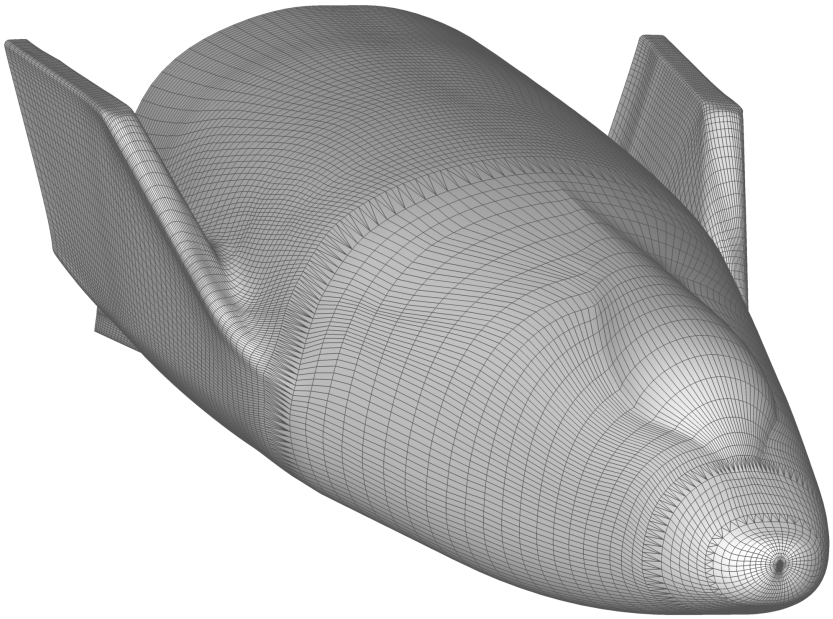


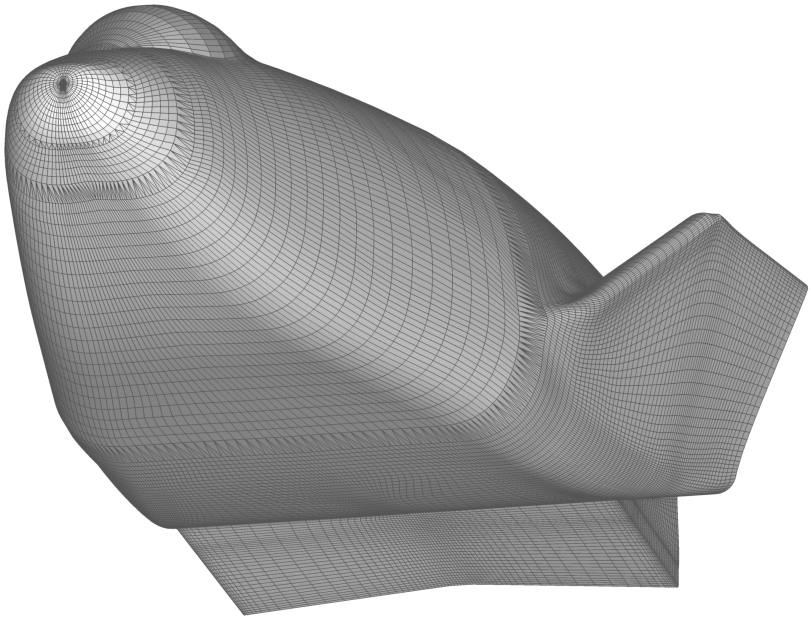
Figure 5.73: Original X-38 single-block structured grid surface discretization.

directly to a reduction in the size of the sparse matrix row which results for the degrees of freedom which lie at this point.

The off-body mesh consists of 60 layers marched hyperbolically from the surface [75], resulting in a hybrid hexahedral/prismatic discretization of the flowfield containing 881,820 elements. The total number of nodes in the mesh is 893,162, which is essentially a factor of two reduction from the original structured grid. The mesh was partitioned and run on 80 processors of the *Columbia* supercomputer located at NASA Ames.



(a) Top view.



(b) Bottom view.

Figure 5.74: X-38 Crew Return Vehicle hybrid element surface mesh. Note the planar lower surface geometry, constant curvature lower-to-upper surface transition, and the deployed, non-planar body flap.

5.6.9.2 Results

The global flowfield about the vehicle is show in Figure 5.75. The strong bow shock/deep leeside expansion of the previous cases is clearly evident at this high angle of attack. The smooth curvature of the sides and upper surface of the vehicle appear to delay

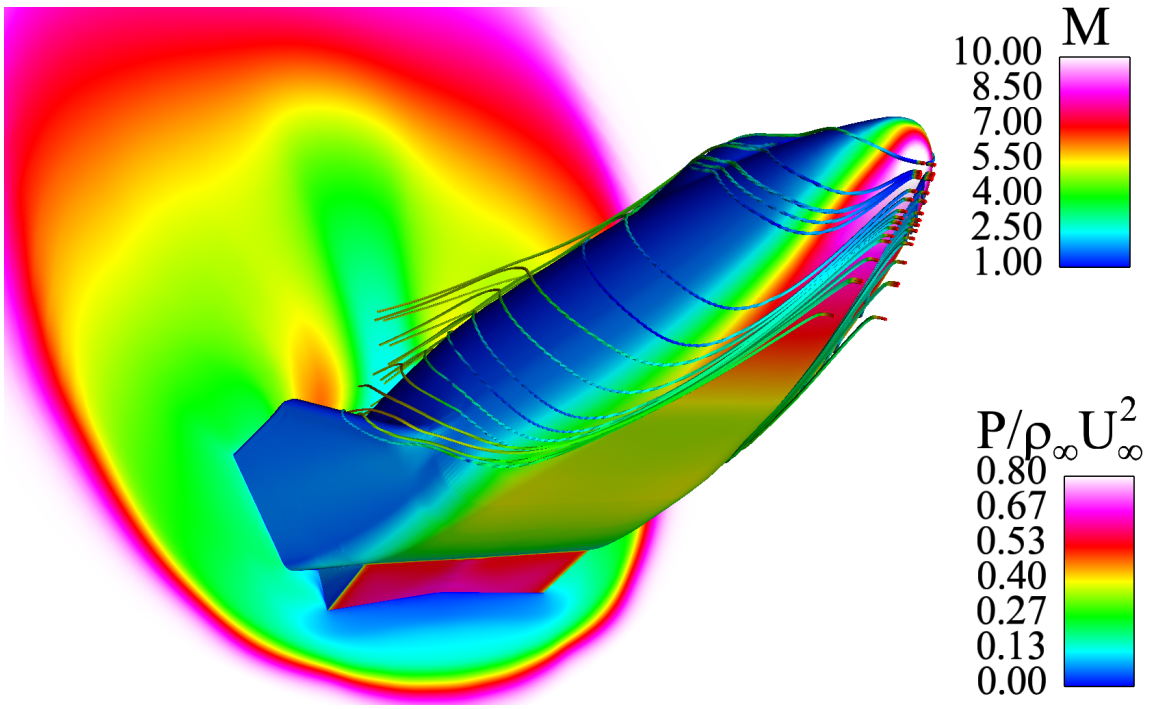


Figure 5.75: X-38 flowfield. Streamlines are colored by local velocity, the surface is colored by static pressure. The cut plane at x/L=0.95 shows the local Mach number.

leeside separation. Recalling Figure 5.64, the strong rotation introduced by the double-delta and abrupt sidewall/payload bay door geometric features of the Space Shuttle Orbiter are absent for this smooth geometry.

The constant axial planar cut at the aft of the vehicle is colored by Mach number. Several interesting flow structures are visible in the figure. The low-speed supersonic flow over the body flap is a result of the attached, oblique shock which is set up at the flap hinge line. As the flow expands around the vehicle into the wake, it accelerates to high Mach

number (on the order of 7 in the fin region). The converging flow then meets at the upper pitch plane and is processed by a weak shock, slowing the flow to Mach 2-5.

The structure of the wake flowfield and upper surface pressure distribution is illustrated more clearly in Figure 5.76. The surface pressure contours are spaced quadratically

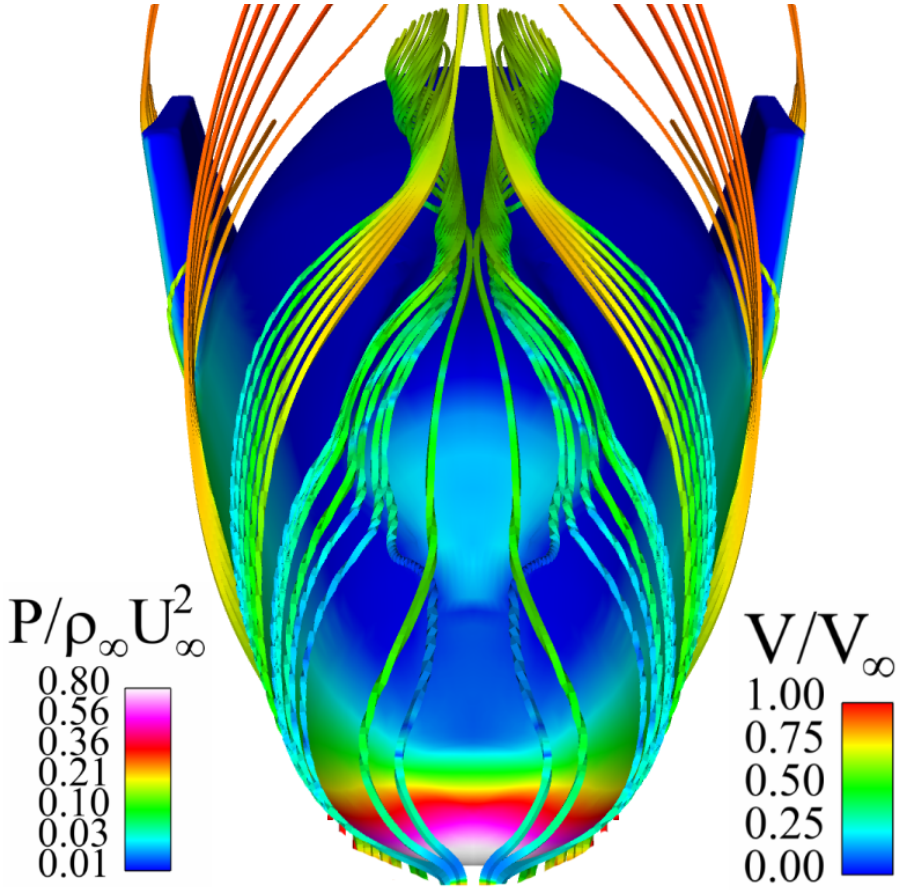


Figure 5.76: X-38 upper surface flowfield. Streamlines are colored by local velocity, the surface is colored by static pressure. The pressure contours are spaced quadratically to illustrate the upper-surface features.

in this case to illustrate the upper surface flow features. The surface pressure is seen to decrease dramatically as the flow expands over the nose of the vehicle. A weak canopy shock forms and induces slightly elevated pressure on the front window housing. The expanding

flow is seen to merge on the upper pitch plane and passes through an oblique shock (as seen in the previous figure). The smoothness of the geometry is seen to delay flow separation well onto the upper surface of the vehicle. When the flow finally does separate a pair of counter-rotating vortices are seen to form behind the canopy.

The elevated pressure on the body flap is clearly evident in Figure 5.75. The details of the flowfield in the region of the body flap are shown in Figure 5.77. Clearly evident in

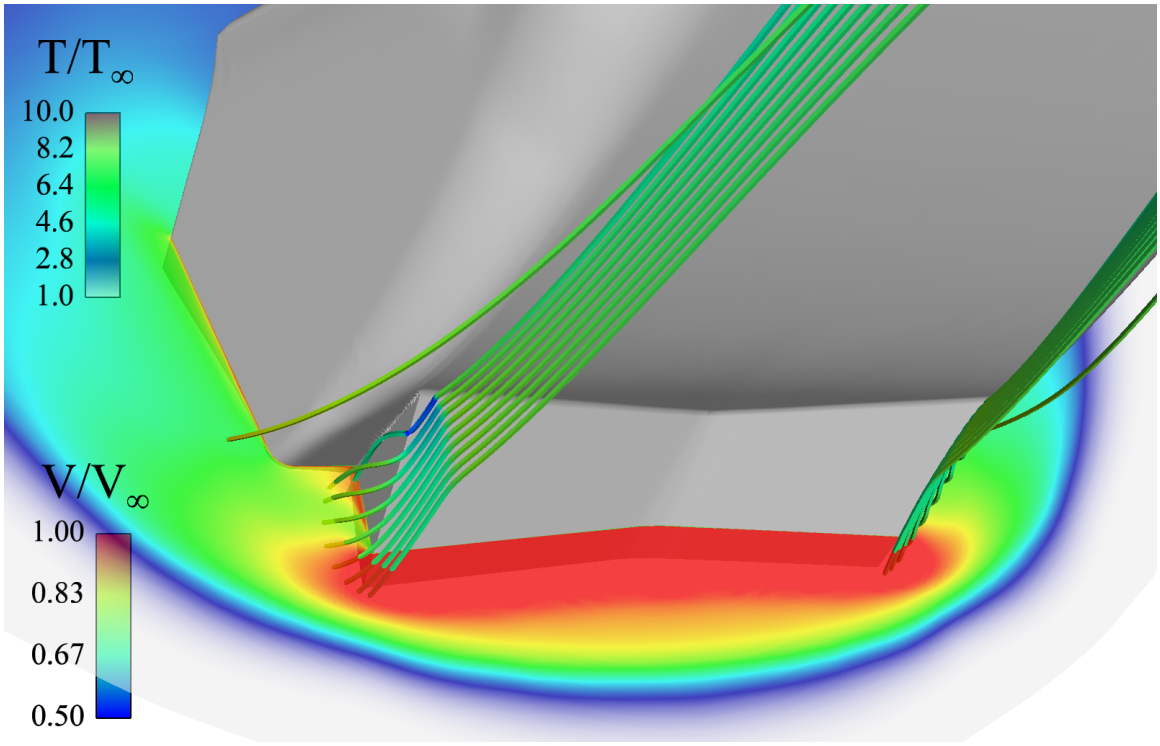


Figure 5.77: X-38 body flap flowfield. Streamlines are colored by velocity, the translucent cut plane depicts static temperature.

the image is the rotation induced by the body flap as the flow expands and "spills over" the side. The pressure on the surface of the flap corresponds to $C_p \approx 1.2$, while vertical side of the flap (as modeled) is at $C_p \approx 0$. This severe, lateral pressure gradient at the side edge of the flap adds a substantial outboard momentum component to the otherwise streamwise flow. The end result is that appreciable vorticity is introduced into the flow in this region.

Figure 5.78 shows the pressure distribution in the pitch plane in the region of the body flap. Vectors located on this plane illustrate the local velocity orientation. The con-

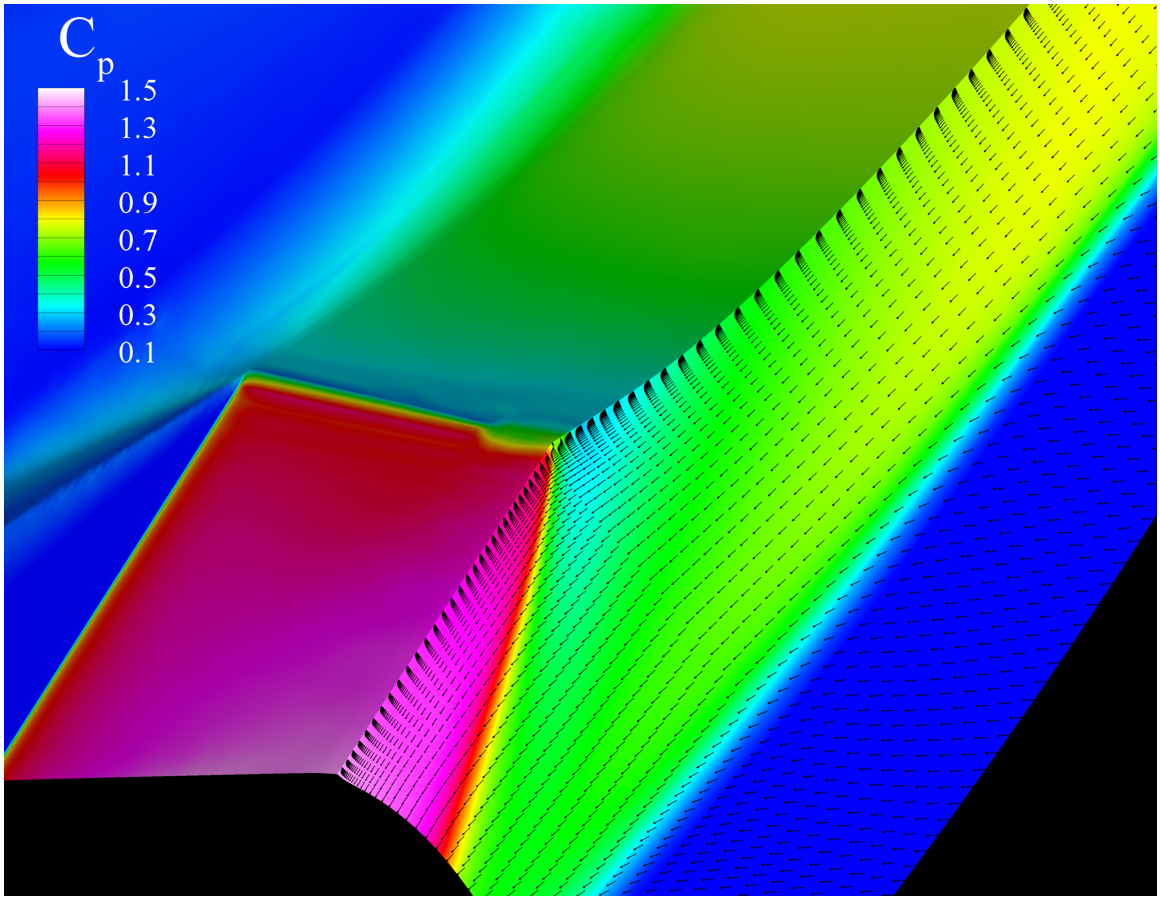


Figure 5.78: X-38 body flap and pitch plane flowfield showing very localized separation.

tours on the vehicle surface correspond to lines of constant pressure and are colored accordingly. Interestingly, there is only a small pressure-induced separation region at the base of the flap. This behavior is somewhat surprising, especially given the flowfield which was observed for the 15° two-dimensional compression ramp. A closer investigation of the flowfield in this region reveals an interesting feature of the vehicle geometry which may contribute to this behavior.

The vehicle outer mold line exhibits substantial convex curvature immediately upstream of the flap hinge line. Upstream of this curvature region the pitch plane is analogous to a flat plate. Therefore, the pressure distribution on the upstream portion of the vehicle in this region is relatively constant. This zero-pressure gradient flow is analogous to the compression ramp studied previously.

In the convex curvature region, however, the flow expands and develops a quite favorable pressure gradient. The result is that the boundary layer profile becomes more full, resulting in higher momentum close to the wall. Such a boundary layer profile is less susceptible to separation. When this relatively full boundary layer encounters the adverse pressure gradient caused by the ramp it is seen to remain largely attached. The flow does exhibit minor separation on the vehicle centerline in the flap dihedral corner, as evident by the elevated pressure in this region. (The important question of mesh resolution will be addressed in Section 5.6.9.3).

It is clear from the figure that the pressure on the centerline portion of the body flap is slightly higher than in the nearby, outboard region. This elevated pressure is a result of the reinforcing mechanisms of three-dimensional edge expansion and the dihedral angle of the flap itself. The split body flap is not flat and, in this case, actually directs flow toward the vehicle centerline. The end result is that the inboard momentum induced by the flap dihedral results in increased pressure as two opposing streams meet on centerline.

The pressure coefficient distribution along the pitch plane is shown in Figure 5.79. For reference, the lower surface cross-section is included in the figure. Interestingly, there is a very slight adverse pressure gradient in the region $0.45<(x/L)<0.6$, even though the vehicle surface is flat. An inspection of Figure 5.75 provides insight into this behavior. It is clear from the figure that the onset of the slight adverse pressure gradient corresponds the location of the lateral fins. The three-dimensional pressure relief caused by the windward-to-leeward expansion is somewhat impeded by the fins, hence the pressure on the lower surface is slightly higher in this vicinity than on the upstream portion of the body.

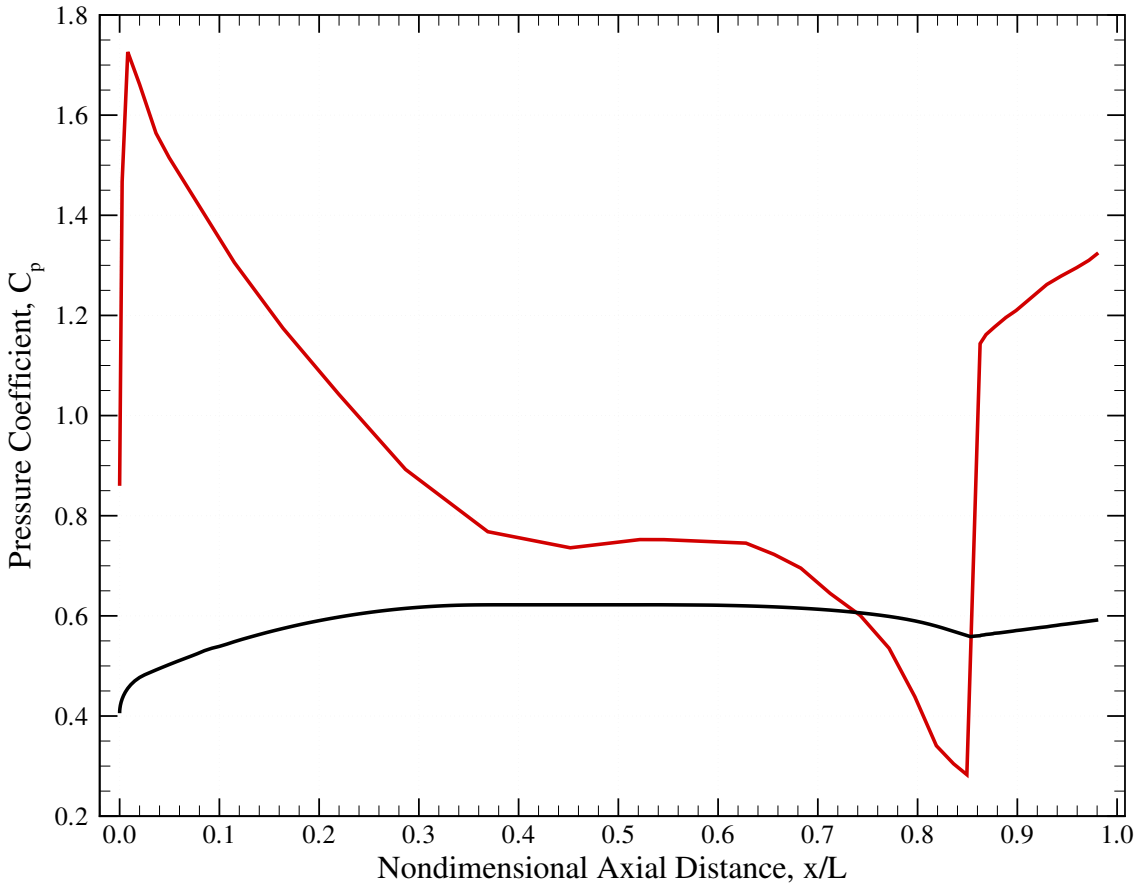


Figure 5.79: X-38 pitch plane lower surface pressure distribution.

The favorable-to-adverse pressure gradient region is clearly shown at (x/L) of approximately 0.85. This convex curvature design feature (and corresponding favorable pressure gradient) apparently mitigates the separation and reattachment interaction which occurs for the case of a compression ramp positioned on a flat plate. This observation will be considered further in the following section.

5.6.9.3 Adaptive Mesh Refinement

The accuracy with which the separated region at the base of the body flap is captured is highly dependent on the adequacy of the mesh in this region. Upon achieving a

steady solution with the baseline mesh, the simulation was continued with the mesh uniformly refined in a specified region of interest. This region contains the body flap and the immediate upstream portion of the vehicle. Specifically, the mesh is uniformly refined in the region

$$(x/L) > 0.8$$
$$(y/L) < 0.2$$
$$-0.2 < (z/L) < -0.02$$

The uniform refinement scheme used in this work locally increases the mesh density by a factor of eight. The resulting mesh contains 1,802,022 nodes and 1,764,205 active elements. The implicit linear system contains 9,010,110 degrees of freedom, and is the largest problem considered in this work. As mentioned previously, the initial mesh was run on 80 processors of the *Columbia* supercomputer at NASA Ames Research Center. The adapted mesh was run on 128 processors of the *lonestar* cluster located at The University of Texas at Austin's Texas Advanced Computing Center.

The aft portion of the vehicle surface and pitch plane is shown in Figure 5.80. The computational mesh is shown on the surface of the vehicle and is colored by static pressure. Additionally, the mesh on the pitch-plane is shown colored by static temperature. The refined portion of the mesh is seen to contain the oblique shock formed by the compression ramp. Additionally, the refined portion of the mesh extends outboard past the edge of the body flap in order to capture the "spill-over" effect discussed previously.

Due to the dihedral angle of the split body flap, the separated region is largest at the vehicle centerline. That is, the incoming flow is directed slightly inboard by the spanwise cant of the body flap. This results in not only a fore-to-aft favorable pressure gradient but also an outboard-to-inboard stabilizing gradient. This lateral flow necessarily terminates on the centerline, hence the separated extent is largest in this region.

The extent of the recirculation region is larger than in the baseline case. Recall from Section 5.6.4 that the effect of increasing Reynolds number is to increase the size of

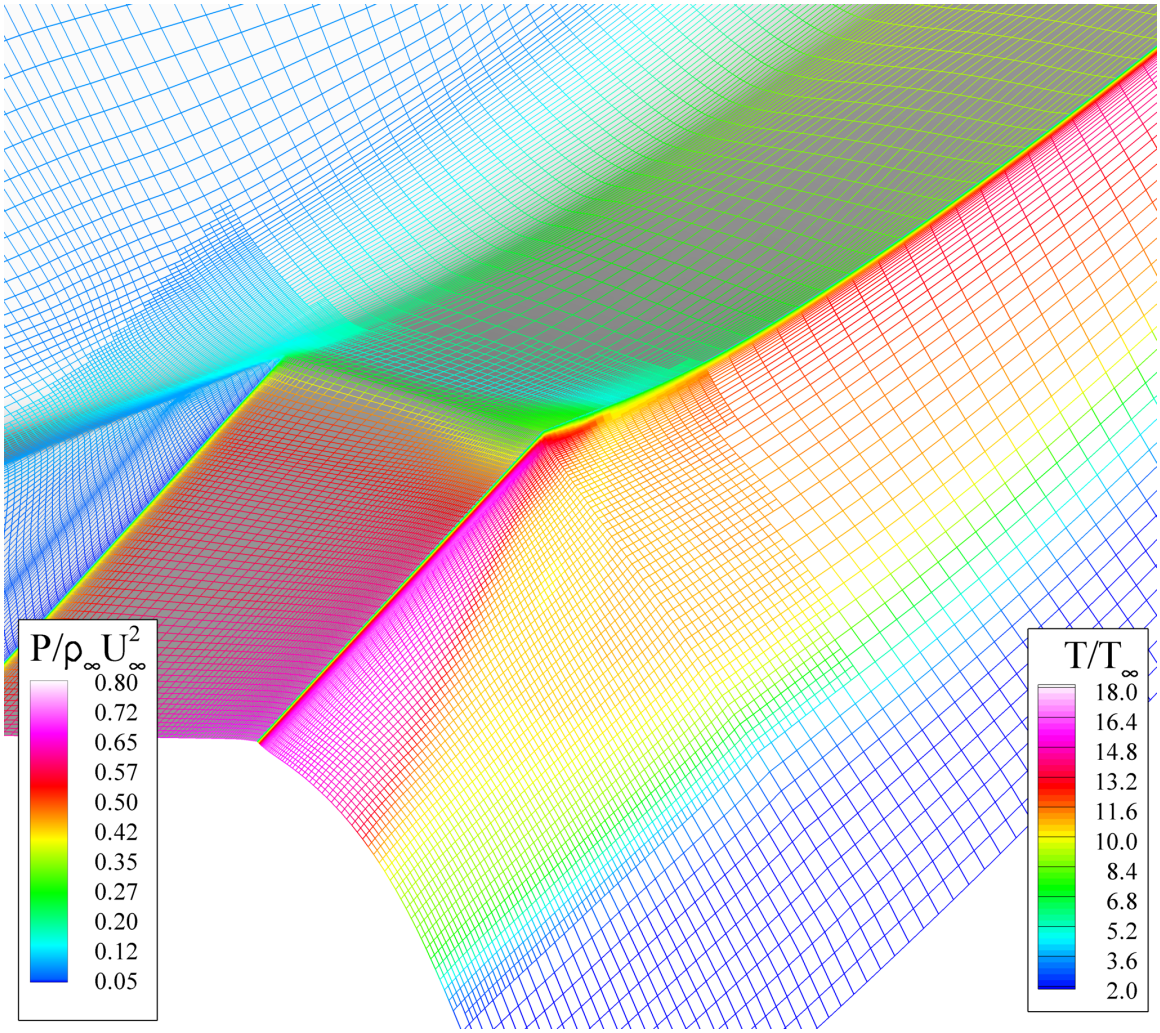


Figure 5.80: X-38 refined mesh. The pitch-plane is colored by static temperature, and the surface is colored by static pressure.

a laminar separated region. Since the baseline and adapted simulations were performed at the same Reynolds number, it seems apparent that increasing the mesh resolution in the separation region (and consequently reducing the numerical dissipation in the scheme) has an effect similar to altering the Reynolds number. From this perspective it is clear that the baseline mesh is not adequately resolved for this case in the separation region.

The pitch-plane pressure coefficient is plotted again in Figure 5.81. The baseline and adapted cases are in excellent agreement for $(x/L) < 0.82$. Beyond this point the

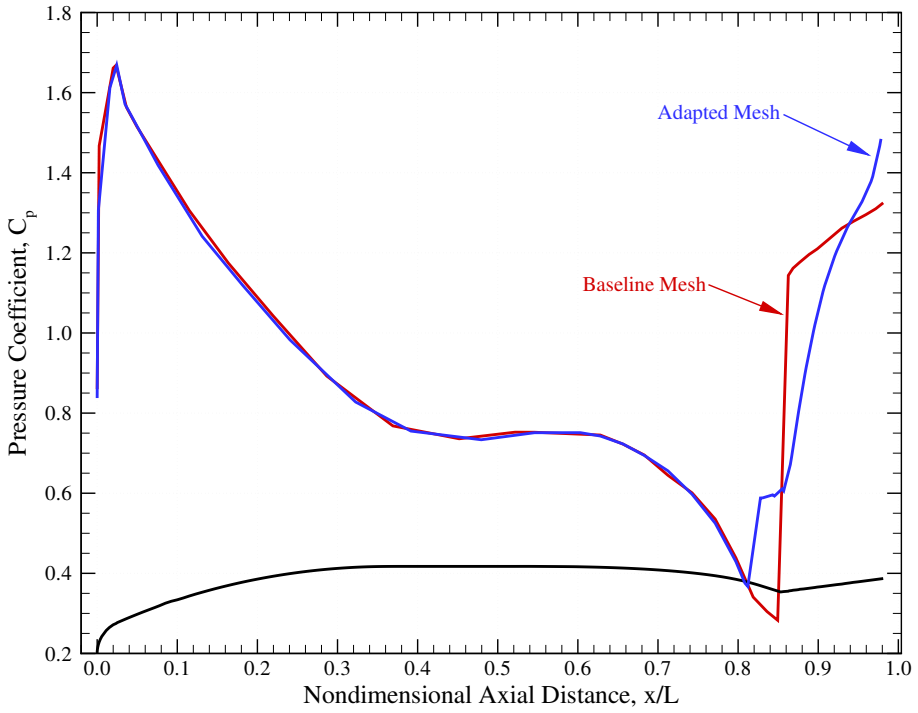


Figure 5.81: X-38 pitch plane lower surface pressure distribution for baseline and adapted meshes.

pressure distributions are markedly different. The rapid increase in pressure for the adapted case corresponds to the onset of separation. The pressure distribution on the body flap is also significantly impacted by the flow reattachment, which occurs further downstream for the adapted mesh case. The difference in the size of the separated region between these two cases demonstrates the sensitivity of this flow phenomenon to accurate mesh resolution.

Chapter 6

Conclusions

Numerical simulations which use adaptive mesh refinement allow for detailed investigations of multiscale phenomena to be performed efficiently. Still, the use of AMR techniques is not widespread in the engineering community. This is particularly the case of simulations performed on massively parallel computers. This work considered the viability of adaptive finite element methods for simulating of flow and transport problems, particularly on high-performance parallel computers.

6.1 Adaptive Mesh Refinement on Parallel Computers

Chapter 2 described a number of key data structures and algorithms which enable adaptive mesh refinement simulations. Object-oriented programming approaches are natural when implementing adaptive methods, particularly in the case of hybrid unstructured meshes, and are overviewed. Key concepts such as degree of freedom constraints, element-independent adaptivity, and tree data structures were reviewed. Error indicators and refinement criteria guide the process of refinement and were discussed.

The dynamic nature of AMR simulations has complicated their efficient implementation on parallel computers. Chapter 3 considered the particular issues that arise when implementing adaptive methods on distributed memory machines and posed particular solutions to a number of these difficulties. The domain-decomposition approach for achieving parallelism in finite element simulations was discussed, along with the important issue of dynamic load balancing which must be addressed in any parallel adaptive software implementation.

The ideas presented in Chapters 2 and 3 have been implemented in the open-source software library `libMesh`. The basis for the library is the observation that the majority of the enabling software technology required by adaptive methods is independent of the class of problems being solved. Thus, by implementing this technology as a set of core services independent of an application code, the considerable level-of-effort required to implement parallel adaptive finite element simulations was amortized across a wide range of applications. The physics-independent software framework `libMesh` was developed during the course of this work and has been adopted by a number of researchers both in the U.S. and abroad.

6.2 Biological Transport

The particular application of chemotactic bacteria systems was addressed in Chapter 4. Such systems have been observed to form very complex, transient patterns. This work applies the developed adaptive technology to the problem of pattern formation in *E.coli* bacteria colonies.

A nonlinear reaction-diffusion set of coupled partial differential equations is presented in detail. This numerical model, first presented by Murray et al. [17], forms the basis for a finite element formulation which is then used to perform a number of application studies. The numerical method used to approximate the mathematical model is discussed in detail. A time integration technique with adaptive temporal error control based on the the Adams-Bashforth predictor/corrector scheme was presented, as was the Newton linearization used to solve the highly nonlinear, coupled set of equations.

The application studies examined the important questions of temporal and mesh convergence. Both the short- and long-term behavior of a chemotactic system is examined in detail. The chapter culminates in the first known application of AMR to the problem of pattern formation in chemotactic biological systems. This work demonstrated the applicability of AMR to such systems and paves the way for detailed parametric studies of chemotactic processes using efficient adaptive techniques.

6.3 Compressible Flows

Chapter 5 treats the problem of hypersonic aerothermodynamics which is of interest in the field of aerospace engineering. Hypersonic flows are characterized by convection-dominated flowfields exhibiting highly localized features such as shock waves and boundary layers. These features make them ideally suited for the adaptive techniques developed in this work.

The compressible Navier-Stokes equations for a calorically perfect, laminar gas for the basic mathematical model used for simulating this class of flows. The equations form a tightly coupled system of nonlinear partial-differential equations which form a mixed elliptic–hyperbolic set for the case of supersonic flows. The governing equations were described in detail. A stabilized weak form was then derived which uses the streamline-upwind Petrov-Galerkin (SUPG) finite element method to simulate high-Reynolds number flows. The SUPG scheme is augmented by a modified shock capturing operator which is required to eliminate spurious oscillations in the vicinity of shock waves.

This stabilized weak formulation was the basis for the discrete finite element model developed in this work. This model uses a novel approach for discretizing the inviscid flux terms which appear in the stabilized formulation. This approach has proven more stable than the traditional method and has enabled the SUPG finite element scheme to be applied in this work to very complex flows.

The performance of the fully implicit, adaptive finite element algorithm was then tested through a number of application studies. These studies include that of inviscid flow about a blunt body and viscous/inviscid interaction in the case of a hypersonic flow over compression corners. Critical issues such as nonlinear solver convergence, temporal convergence, and mesh convergence were all addressed. The suitability of AMR for this class of problems was also investigated in the context of these benchmark cases. The method was validated by comparison to experimentally-measured quantities of interest such as surface pressure, shear, and heat transfer distributions.

The validated method was then applied to a number of complex applications including hypersonic flow about a double cone, periodic flow about a spherical nose tip/cavity configuration, and type IV shock/shock interaction on a cylinder. Experimental data, including heat transfer distributions and optically-measured flowfield properties, were used to provide additional validation for these complex cases. Finally, the chapter culminates in the investigation of hypersonic flow about a number of complex three-dimensional geometries (including the Space Shuttle Orbiter) at reentry conditions. The adaptive mesh refinement technology developed in this work is applied in three dimensions to the case of the X-38 Crew Return Vehicle at wind tunnel conditions. The extent of the separated region resulting from the deflected body flap is seen to be quite sensitive to the mesh resolution employed in the simulation.

While only laminar, calorically perfect gases are considered in this work, the AMR technology is expected to generalize directly to the case of turbulent and/or reacting flows. Future work will extend the range of applicability of the finite element model by including state equations for gases in thermal equilibrium. The effects of turbulence may be included through the typical Reynolds-Averaged Navier-Stokes approach by implementing suitable turbulence models.

This work examined highly dynamic flows which result due to both freestream noise in conventional wind tunnels and because of natural instabilities in the complex flowfields. Favorable comparisons were found with average values measured in experiments. Further work employing high-rate data measurement techniques with fast response surface instrumentation could be used to further validate the complex transient flows seen in some parts of this work.

Shock waves are largely one-dimensional structures in that they exhibit extremely high gradients in predominantly one direction. As such, the isotropic mesh refinement approach used in this work may lead to a proliferation of degrees of freedom. This is especially the case in three dimensions. Future work should consider coupling a mesh

redistribution/smoothing scheme with anisotropic refinement so that these features may be resolved optimally with a limited set of resources.

The streamline-upwind Petrov-Galerkin finite element formulation used here is applicable on fully unstructured meshes. Future work should apply the current finite element scheme to a range of unstructured mesh topologies in order to assess the influence of mesh quality in aerothermodynamic applications During the course of this work this capability was just barely exercised (recall the hybrid surface discretization of figure 5.74). This was largely due to the relatively simple geometries used for validation and the availability of preexisting block-structured grids. The use of unstructured meshes in the context of aerothermodynamic applications is still in its infancy, largely because to date the majority of finite volume schemes applied in the unstructured setting have produced unsatisfactory results. Still, the ability to use unstructured meshes (as opposed to block-structured grids) is highly desirable because the time associated with mesh generation may be drastically reduced. The SUPG scheme developed in this work provides a natural way to assess the influence of unstructured mesh quality on aerothermodynamic predictive capability.

Appendix

Appendix A

Compressible Flow

A.1 Jacobian Matrices

In the following sections several parameters (in addition to those described in Section 5.2) are used for notational convenience:

$$
\begin{aligned}
q^2 &= u^2 + v^2 + w^2 \\
H &= E + \frac{P}{\rho} \\
\nu &= \frac{\mu}{\rho} \\
\overline{\nu} &= \nu - \frac{k}{\rho c_v} \\
\vartheta &= \frac{k}{\rho c_v}\left(\frac{q^2}{2} - e\right) - \nu q^2
\end{aligned}
$$

A.1.1 Inviscid Flux Jacobians

Recall Equation (5.32):

$$
\frac{\partial \boldsymbol{F}_i}{\partial x_i} = \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{U}} \frac{\partial \boldsymbol{U}}{\partial x_i} = \boldsymbol{A}_i \frac{\partial \boldsymbol{U}}{\partial x_i}
$$

where $\boldsymbol{A}_i = \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{U}}$ is the inviscid flux Jacobian. Further, since $\boldsymbol{F}_i$ is a homogeneous function of degree one the following holds:

$$
\boldsymbol{F}_i = \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{U}} \boldsymbol{U} = \boldsymbol{A}_i \boldsymbol{U}
$$

The particular $\boldsymbol{A}_i$'s for the conserved variables $\boldsymbol{U} = [\rho, \rho u, \rho v, \rho w, \rho E]^T$ are explicitly defined as

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{\gamma-1}{2}q^2 - u^2 & (3-\gamma)\,u & (1-\gamma)\,v & (1-\gamma)\,w & \gamma-1 \\ -uv & v & u & 0 & 0 \\ -uw & w & 0 & u & 0 \\ \left(\frac{\gamma-1}{2}q^2 - H\right)u & H + (1-\gamma)\,u^2 & (1-\gamma)\,uv & (1-\gamma)\,uw & \gamma u \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ -uv & v & u & 0 & 0 \\ \frac{\gamma-1}{2}q^2 - v^2 & (1-\gamma)\,u & (3-\gamma)\,v & (1-\gamma)\,w & \gamma-1 \\ -vw & 0 & w & v & 0 \\ \left(\frac{\gamma-1}{2}q^2 - H\right)v & (1-\gamma)\,uv & H + (1-\gamma)\,v^2 & (1-\gamma)\,vw & \gamma v \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ -uw & w & 0 & u & 0 \\ -vw & 0 & w & v & 0 \\ \frac{\gamma-1}{2}q^2 - w^2 & (1-\gamma)\,u & (1-\gamma)\,v & (3-\gamma)\,w & \gamma-1 \\ \left(\frac{\gamma-1}{2}q^2 - H\right)w & (1-\gamma)\,uw & (1-\gamma)\,vw & H + (1-\gamma)\,w^2 & \gamma w \end{bmatrix}$$

A.1.2 Viscous Flux Jacobians

$$K_{11} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -\frac{4}{3}\nu u & \frac{4}{3}\nu & 0 & 0 & 0 \\ -\nu v & 0 & \nu & 0 & 0 \\ -\nu w & 0 & 0 & \nu & 0 \\ \vartheta - \frac{1}{3}\nu u^2 & \left(\frac{1}{3}\nu + \overline{\nu}\right)u & \overline{\nu}v & \overline{\nu}w & \frac{k}{\rho c_v} \end{bmatrix}$$

$$K_{12} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{2}{3}\nu v & 0 & -\frac{2}{3}\nu & 0 & 0 \\ -\nu u & \nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{3}\nu u v & \nu v & -\frac{2}{3}\nu u & 0 & 0 \end{bmatrix}$$

$$K_{13} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{2}{3}\nu w & 0 & 0 & -\frac{2}{3}\nu & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\nu u & \nu & 0 & 0 & 0 \\ -\frac{1}{3}\nu u w & \nu w & 0 & -\frac{2}{3}\nu u & 0 \end{bmatrix}$$

$$K_{21} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -\nu v & 0 & \nu & 0 & 0 \\ \frac{2}{3}\nu u & -\frac{2}{3}\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{3}\nu v u & -\frac{2}{3}\nu v & \nu u & 0 & 0 \end{bmatrix}$$

$$K_{22} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -\nu u & \nu & 0 & 0 & 0 \\ -\frac{4}{3}\nu v & 0 & \frac{4}{3}\nu & 0 & 0 \\ -\nu w & 0 & 0 & \nu & 0 \\ \vartheta - \frac{1}{3}\nu v^2 & \overline{\nu} u & \left(\frac{1}{3}\nu + \overline{\nu}\right) v & \overline{\nu} w & \frac{k}{\rho c_v} \end{bmatrix}$$

$$K_{23} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{2}{3}\nu w & 0 & 0 & -\frac{2}{3}\nu & 0 \\ -\nu v & 0 & \nu & 0 & 0 \\ -\frac{1}{3}\nu v w & 0 & \nu w & -\frac{2}{3}\nu v & 0 \end{bmatrix}$$

$$K_{31} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -\nu w & 0 & 0 & \nu & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{2}{3}\nu u & -\frac{2}{3}\nu & 0 & 0 & 0 \\ -\frac{1}{3}\nu w u & -\frac{2}{3}\nu w & 0 & \nu u & 0 \end{bmatrix}$$

$$K_{32} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\nu w & 0 & 0 & \nu & 0 \\ \frac{2}{3}\nu v & 0 & -\frac{2}{3}\nu & 0 & 0 \\ -\frac{1}{3}\nu w v & 0 & -\frac{2}{3}\nu w & \nu v & 0 \end{bmatrix}$$

$$K_{33} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -\nu u & \nu & 0 & 0 & 0 \\ -\nu v & 0 & \nu & 0 & 0 \\ -\frac{4}{3}\nu w & 0 & 0 & \frac{4}{3}\nu & 0 \\ \vartheta - \frac{1}{3}\nu w^2 & \overline{\nu} u & \overline{\nu} v & \left(\frac{1}{3}\nu + \overline{\nu}\right) w & \frac{k}{\rho c_v} \end{bmatrix}$$

A.2 Conservation Variables to Entropy Variables Transformation

Let $\boldsymbol{V}$ be the so-called entropy variables for the compressible Navier-Stokes system considered in Chapter 5. Specifically,

$$\boldsymbol{V} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{bmatrix} \equiv \frac{1}{\rho e} \begin{bmatrix} -\rho E + \rho e \left(\gamma - s + 1\right) \\ \rho u \\ \rho v \\ \rho w \\ -\rho \end{bmatrix}$$

The transformation from the conservation variables $\boldsymbol{U}$ to the entropy variables $\boldsymbol{V}$ is defined such that

$$\boldsymbol{V} = \boldsymbol{A}_0 \, \boldsymbol{U}$$

The inverse of this transformation,

$$\boldsymbol{U} = \boldsymbol{A}_0^{-1} \, \boldsymbol{V}$$

is used in computing the shock capturing operator δ (c.f. Equation 5.42). This symmetric matrix is given by

$$A_0^{-1} = \begin{bmatrix} k^2+\gamma & kV_2 & kV_3 & kV_4 & (k+1)\,V_5 \\ kV_2 & V_2^2 - V_5 & V_2V_3 & V_2V_4 & V_2V_5 \\ kV_3 & V_3V_2 & V_3^2 - V_5 & V_3V_4 & V_3V_5 \\ kV_4 & V_4V_2 & V_4V_3 & V_4^2 - V_5 & V_4V_5 \\ (k+1)\,V_5 & V_5V_2 & V_5V_3 & V_5V_4 & V_5^2 \end{bmatrix}$$

where $k = \left(\frac{V_2^2+V_3^2+V_4^2}{2V_5}\right)$, and the entropy is defined as $s = \ln\left[\frac{(\gamma-1)\rho e}{\rho^\gamma}\right]$.

A.3 Rankine–Hugoniot Jump Conditions

To determine the change in flowfield properties across a stationary normal shock wave it is sufficient to consider the steady one–dimensional Euler equations:

$$\frac{\partial}{\partial x}\begin{bmatrix} \rho u \\ \rho u^2 + P \\ (\rho E + P)\,u \end{bmatrix} = 0 \tag{A.1}$$

Denoting the pre and post–shock conditions by $()_1$ and $()_2$, respectively, Equation (A.1) implies

$$\rho_1 u_1 = \rho_2 u_2 \tag{A.2}$$

$$\rho_1 u_1^2 + P_1 = \rho_2 u_2^2 + P_2 \tag{A.3}$$

$$(\rho_1 E_1 + P_1)\,u_1 = (\rho_2 E_2 + P_2)\,u_2 \tag{A.4}$$

From (A.2) we define κ, the ratio of the upstream to downstream density:

$$\kappa \equiv \frac{\rho_1}{\rho_2} = \frac{u_2}{u_1} \tag{A.5}$$

For a calorically perfect gas (see (5.15)–(5.16)) the following holds

$$\rho E = \frac{P}{\gamma - 1} + \frac{1}{2}\rho u^2 \tag{A.6}$$

which, upon substitution into (A.4) yields

$$\begin{aligned}
\left(\frac{\gamma}{\gamma-1}P_1+\frac{1}{2}\rho_1 u_1^2\right)u_1 &= \left(\frac{\gamma}{\gamma-1}P_2+\frac{1}{2}\rho_2 u_2^2\right)u_2 \\
\left(\frac{\gamma}{\gamma-1}\frac{P_1}{\rho_1}+\frac{1}{2}u_1^2\right)\rho_1 u_1 &= \left(\frac{\gamma}{\gamma-1}\frac{P_2}{\rho_2}+\frac{1}{2}u_2^2\right)\rho_2 u_2 \\
\frac{\gamma}{\gamma-1}\frac{P_1}{\rho_1}+\frac{1}{2}u_1^2 &= \frac{\gamma}{\gamma-1}\frac{P_2}{\rho_2}+\frac{1}{2}u_2^2
\end{aligned} \tag{A.7}$$

Multiplying (A.3) by $\frac{\gamma}{\gamma-1}\frac{1}{\rho_2}$ gives

$$\begin{aligned}
\frac{\gamma}{\gamma-1}\left(\frac{\rho_1 u_1^2}{\rho_2}+\frac{P_1}{\rho_2}\right) &= \frac{\gamma}{\gamma-1}\left(u_2^2+\frac{P_2}{\rho_2}\right) \\
\frac{\gamma}{\gamma-1}\left(u_1^2+\frac{P_1}{\rho_1}\right)\kappa &= \frac{\gamma}{\gamma-1}\left(u_2^2+\frac{P_2}{\rho_2}\right)
\end{aligned}$$

which, when subtracted from (A.7), becomes

$$\begin{aligned}
\frac{\gamma}{\gamma-1}\frac{P_1}{\rho_1}(1-\kappa)+\left(\frac{1}{2}-\frac{\gamma}{\gamma-1}\kappa\right)u_1^2 &= \frac{\gamma+1}{2(1-\gamma)}u_2^2 \\
\frac{\gamma}{\gamma-1}\frac{P_1}{\rho_1}(1-\kappa)+\left(\frac{1}{2}-\frac{\gamma}{\gamma-1}\kappa\right)u_1^2 &= \frac{\gamma+1}{2(1-\gamma)}u_1^2\kappa^2
\end{aligned} \tag{A.8}$$

Recognizing that for a calorically perfect ideal gas $\gamma P/\rho = c^2$ and multiplying (A.8) by $(\gamma-1)/u_1^2$ produces the following quadratic equation:

$$\frac{\gamma+1}{2}\kappa^2-\left(\gamma+\frac{1}{M_1^2}\right)\kappa+\frac{\gamma-1}{2}+\frac{1}{M_1^2}=0$$

which can be factored as

$$(\kappa-1)\left(\frac{\gamma+1}{2}\kappa-\frac{\gamma-1}{2}-\frac{1}{M_1^2}\right)=0 \tag{A.9}$$

The trivial solution $\kappa = 1$ corresponds to no shock wave, i.e. $()_1 = ()_2$. For the case of a shock wave Equation (A.9) predicts

$$\kappa=\frac{\rho_1}{\rho_2}=\frac{u_2}{u_1}=\frac{\gamma-1}{\gamma+1}+\frac{2}{M_1^2(\gamma+1)} \tag{A.10}$$

Equation (A.10) can be used in (A.2)–(A.4) to completely determine the post–shock conditions $()_2$.

Bibliography

[1] Benjamin S. Kirk, John W. Peterson, Roy H. Stogner, and Graham F. Carey. `libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers*, 22(3):237–254, 2006.

[2] G. F. Carey, W. Barth, J. A. Woods, B. S. Kirk, M. L. Anderson, S. Chow, and W. Bangerth. Modelling error and constitutive relations in simulation of flow and transport. *International Journal for Numerical Methods in Fluids*, 46:1211–1236, 2004.

[3] G. F. Carey, R. McLay, W. Barth, S. Swift, and B. Kirk. Distributed Parallel Simulation of Surface Tension Driven Viscous Flow and Transport Processes. In Eduardo Ramos, Gerardo Cisneros, Rafael Fernández-Flores, and Alfredo Santillán-González, editors, *Computational Fluid Dynamics: Proceedings of the Fourth UNAM Supercomputing Conference*, pages 143–155. World Scientific, June 2000.

[4] B. Kirk, K. Lipnikov, and G. F. Carey. Nested Grid Iteration for Incompressible Viscous Flow and Transport. *International Journal of Computational Fluid Dynamics*, 17(4):253–262, August 2003.

[5] G. F. Carey, M. Anderson, B. Carnes, and B. Kirk. Some aspects of adaptive grid technology related to boundary and interior layers. *J. Comput. Appl. Math.*, 166(1):55–86, 2004.

[6] G. F. Carey. *Computational Grids: Generation, Adaptation, and Solution Strategies*. Taylor & Francis, 1997.

[7] James R. Stewart and H. Carter Edwards. A framework approach for developing parallel adaptive multiphysics applications. *Finite Elementsin Analysis and Design*, 40(12):1599–1617, 2004.

[8] Wolfgang Bangerth, Ralf Hartmann, and Guido Kanschat. deal.II — a general purpose object oriented finite element library. *ACM Trans. Math. Software*. to appear.

[9] Eric B. Becker, Graham F. Carey, and J. Tinsley Oden. *Finite Elements – An Introduction*, volume 1. Prentice Hall, 1981.

[10] D. W. Kelly, J. P. Gago, O. C. Zienkiewicz, and I. Babuska. A posteriori error analysis and adaptive processes in the finite element method: Part I Error analysis. *Int. J. Num. Meth. Engng.*, 19:1593–1619, 1983.

[11] M.J. Aftosmis and M.J. Berger. Multilevel Error Esimation and Adaptive h-Refinement for Cartesian Meshes with Embedded Boundaries. 40^{th} AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2002-0863, January 2002.

[12] S. Iqbal and G. F. Carey. Performance analysis of dynamic load balancing algorithms with variable number of processors. *Journal of Parallel and Distributed Computing*, 65(8):934–948, 2005.

[13] Karen Devine, Erik Boman, Robert Heaphy, Bruce Hendrickson, and Courtenay Vaughan. Zoltan data management services for parallel dynamic applications. *Computing in Science and Engineering*, 4(2):90–97, 2002.

[14] G. Karypis and V. Kumar. METIS unstructured graph partitioning and sparse matrix order. Technical report, University of Minnesota, Department of Computer Science, August 1995.

[15] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.3.0, Argonne National Laboratory, April 2004.

[16] E. O. Budrene and H. C. Berg. Dynamics of formation of symmetrical patterns by chemotactic bacteria. *Nature*, 376:49–53, 1995.

[17] J. D. Murray, J. Cook, R. Tyson, and S. R. Lubkin. Spatial Pattern Formation in Biology: I. Dermal Wound Healing. II. Bacterial Patterns. *Journal of the Franklin Institute*, 335(2):303–332, 1998.

[18] E. O. Budrene and H. C. Berg. Complex patterns formed by motile cells of *escherichia coli*. *Nature*, 349:630–633, 1991.

[19] A. J. Staelens. Numerical solutions of a class of nonlinear reaction-diffusion PDE systems. Master's thesis, The University of Texas at Austin, 2002.

[20] A. Iserles. *A first course in the numerical analysis of differential equations*. Cambridge University Press, 1996.

[21] D. E. Woodward, R. Tyson, M. R. Myerscough, J. D. Murray, E. O. Budrene, and H. C. Berg. Spatio-temporal Patterns Generated by Salmonella Typhimurium. *Biophysical Journal*, 68:2181–2189, 1995.

[22] John D. Anderson, Jr. *Modern Compressible Flow: With Histroical Perspective*. McGraw Hill, 2nd edition, 1990.

[23] John D. Anderson, Jr. *Hypersonic and High Temperature Gas Dynamics*. AIAA, Reston, Virginia, 2000.

[24] Ronald L. Panton. *Incompressible Flow*. John Wiley & Sons, 2nd edition, 1996.

[25] J. C. Tannehill and P. H. Mugge. Improved curve fits for the thermodynamic properties of equilibrium air suitable for numerical computation using time-dependent or shock-capturing methods. Technical Report NASA CR-2470, National Aeronautics and Space Administration, October 1974.

[26] Frank M. White. *Viscous Fluid Flow*. McGraw Hill, 2nd edition, 1991.

[27] R. Gupta, J. Yos, R. Thompson, and K. Lee. A Review of Reaction Rates and Thermodynamic and Transport Properties for an 11-Species Air Model for Chemical and Thermal Nonequilibrium Calculations to 30,000 K. Technical Report RP–1232, NASA, August 1990.

[28] J. C. Tannehill, D. A. Anderson, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. Taylor & Francis, Washington, D.C., 2nd edition, 1997.

[29] M. P. Kessler and A. M. Awruch. Analysis of hypersonic flows using finite elements with Taylor–Galerkin scheme. *International Journal for Numerical Methods in Fluids*, 44:1355–1376, 2004.

[30] B.N. Jiang and G.F. Carey. A Stable Least-Squares Finite Element Method for Nonlinear Hyperbolic Problems. *International Journal for Numerical Methods in Fluids*, 8:933–942, 1988.

[31] B.N. Jiang and G.F. Carey. Least-Squares Finite Element Methods for Compressible Euler Equations. *International Journal for Numerical Methods in Fluids*, 10:557–568, 1990.

[32] T. J. R. Hughes, L. P. Franca, and G. M. Hullbert. A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least–squares method advective–diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73:173–189, 1989.

[33] T. J. R. Hughes and M. Mallet. A new finite element formulation for computational fluid dynamics: III. the generalized streamline operator for multidimensional advective–diffusive systems. *Computer Methods in Applied Mechanics and Engineering*, 58:305–328, 1986.

[34] S. K. Aliabadi. *Parallel Finite Element Computations in Aerospace Applications*. PhD thesis, The University of Minnesota, 1994.

[35] S. K. Aliabadi and T. E. Tezduyar. Parallel Fluid Dynamics Computations in Aerospace Applications. *International Journal for Numerical Methods in Fluids*, 21:783–805, 1995.

[36] G. J. LeBeau. The Finite Element Computation of Compressible Flows. Master's thesis, The University of Minnesota, 1990.

[37] T. J. R. Hughes and M. Mallet. A new finite element formulation for computational fluid dynamics: IV. a discontinuity operator for multidimensional advective–diffusive systems. *Computer Methods in Applied Mechanics and Engineering*, 58:329–336, 1986.

[38] Farzin Shakib, Thomas J. R. Hughes, and Zdeněk Johan. A new finite element formulation for computational fluid dynamics: X. the compressible Euler and Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 89:141–219, 1991.

[39] G. Hauke and T. J. R. Hughes. A comparative study of different sets of variables for solving compressible and incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 153:1–44, 1998.

[40] Daryl Lawrence Bonhaus. *A Higher Order Accurate Finite Element Method for Viscous Compressible Flows*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, November 1998.

[41] C. A. J. Fletcher. The Group Finite Element Formulation. *Computer Methods in Applied Mechanics and Engineering*, 37:225–243, 1983.

[42] D. Kuzmin, M. Möller, and S. Turek. High–Resolution FEM–FCT Schemes for Multidimensional Conservation Laws. *Computer Methods in Applied Mechanics and Engineering*, 193:4915–4946, May 2004.

[43] D. Kuzmin and S. Turek. High–Resolution FEM–TVD Schemes Based on a Fully Multidimensional Flux Limiter. *Journal of Computational Physics*, 198:131–158, 2004.

[44] L. Catabriga and A. L. G. A. Coutinho. Improving convergence to steady state of implicit SUPG solution of Euler equations. *Communications in Numerical Methods in Engineering*, 18(5):345–353, May 2002.

[45] Michael D. Greenberg. *Foundations of Applied Mathematics*. Prentice-Hall, 1978.

[46] Zdeněk Johan, Thomas J. R. Hughes, and Farzin Shakib. A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite element analysis in fluids. *Computer Methods in Applied Mechanics and Engineering*, 87:281–304, 1991.

[47] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, Jack Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* Philadelphia: Society for Industrial and Applied Mathematics. Also available as postscript file on http://www.netlib.orgtemplatesTemplates.html, 1994.

[48] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[49] Eric J. Nielsen and Bil Kleb. Efficient Construction of Discrete Adjoint Operators on Unstructured Grids by Using Complex Variables. *AIAA Journal*, 44(4):827–836, April 2006.

[50] Youcef Saad and Martin H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[51] G. Jiang and C.-W. Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126:202–228, 1996.

[52] C.-W. Shu. High order finite difference and finite volume WENO schemes and discontinuous Galerkin methods for CFD. Technical Report ICASE Report No. 2001-11, Brown University, Providence, Rhode Island, May 2001.

[53] Alphonso Ambrosio and Andrzej Wortman. Stangation Point Shock Detachment Distance for Flow Around Spheres and Cylinders. *American Rocket Society*, 32(2):285–287, February 1962.

[54] P. J. Capon. *Adaptive Stable Finite Element Methods for the Compressible Navier-Stokes Equations*. PhD thesis, The University of Leeds, 1995.

[55] Ames Research Staff. Equations, tables, and charts for compressible flow. Technical Report NACA 1135, National Advisory Committee for Aeronautics, 1953.

[56] William L. Barth. *Simulation of non-Newtonian fluids on workstation clusters*. PhD thesis, The University of Texas at Austin, 2004.

[57] Michael S. Holden. A Study of Flow Separation in Regions of Shock Wave-Boundary Layer Interaction in Hypersonic Flow. 11th Fluid and Plasma Dynamics Conference, AIAA Paper 1978-1169, July 1978.

[58] Michael S. Holden. Historical Review of Experimental Studies and Prediction Methods to Describe Laminar and Turbulent Shock Wave/Boundary Layein Interactions in Hypersonic Flows. 44th AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2006-494, January 2006.

[59] R. Abgrall, J.-A. Désidéri, R. Glowinski, M. Mallet, and J. Périaux. *Hypersonic flows for reentry problems*, volume 3. Springer-Verlag, 1991.

[60] Randolph P. Lillard and Kevin M. Dries. Laminar Heating Validation of the OVERFLOW Code. 43rd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2005-685, January 2005.

[61] Timothy P. Wadhams and Michael S. Holden. Summary of Experimental Studies for Code Validation in the LENS Facility and Comparisons with Recent Navier-Stokes and DSMC Solutions for Two- and Three-dimensional Separated Regions in Hypervelocity Flows. 42nd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2004-917, January 2004.

[62] Peter A. Gnoffo. Validation studies for hypersonic flow prediction. 39th AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2001-1025, January 2001.

[63] Matthew MacLean and Michael Holden. Validation and comparison of WIND and DPLR results for hypersonic, laminar problems. 42nd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2004-529, January 2004.

[64] Ioannis Nompelis, Graham V. Candler, and Michael S. Holden. Effect of Vibrational Nonequilibrium on Hypersonic Double-Cone Experiments. *AIAA Journal*, 41(11):2162–2169, November 2003.

[65] Joseph J. Coblish, Michael S. Smith, Terrell Hand, Graham V. Candler, and Ioannis Nompelis. Double-Cone Experiment and Numerical Analysis at AEDC Hypervelocity Wind Tunnel No. 9. 43rd AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2005-0902, January 2005.

[66] Michael S. Holden and Timothy P. Wadhams. A Database Of Aerothermal Measurements In Hypersonic Flow In "Building Block" Experiments for CFD Validation. 41st AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 2003-1137, January 2003.

[67] Joseph Olejniczak. *Computational and Experimental Study of Nonequilibrium Chemsitry in Hypersonic Flows*. PhD thesis, The University of Minnesota, April 1997.

[68] Emmanuel Villermaux. On the role of viscosity in shear instabilities. *Physics of Fluids*, 10(2):368–373, February 1998.

[69] John F. Lafferty and Joseph D. Norris. Measurements of Fluctuating Pitot Pressure, "Tunnel Noise," in the AEDC Hypervelocity Wind Tunnel No. 9. AIAA Paper 2007-1678, February 2007.

[70] W. A. Engblom, D. B. Goldstein, D. Landoon, and S. P. Schneider. Fluid dynamics of hypersonic forward-facing cavity flow. 34th AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 1996-667, January 1996.

[71] W. A. Engblom and D. B. Goldstein. Nose-tip surface heat reduction mechanism. 34th AIAA Aerospace Sciences Meeting and Exhibit, AIAA Paper 1996-354, January 1996.

[72] Sidra I. Silton and David B. Goldstein. Ablation Onset in Unsteady Hypersonic Flow About Nose Tip with Cavity. *Journal of Thermophysics and Heat Transfer*, 14(3):421–434, July–September 2000.

[73] Sidra Idelle Silton. *Ablation onset in unsteady hypersonic flow about nose-tips with a forward-facing cavity.* PhD thesis, The University of Texas at Austin, 2001.

[74] Sidra I. Silton and David B. Goldstein. Use of an Axial Nose-Tup Cavity for Delaying Ablation Onset in Hypersonic Flow. *Journal of Fluid Mechanics*, 528:297–321, 2005.

[75] John P. Steinbrenner and John R. Chawner. Gridgen's Implementation of Partial Differential Equation Based Structured Grid Generation Methods. In *IMR*, pages 143–152, 1999.

[76] S. Yoon and A. Jameson. Lower–Upper Implicit Schemes with Multiple Grids for the Euler Equations. *AIAA Journal*, 25:929–935, 1987.

[77] S. Yoon and A. Jameson. Lower–Upper Symmetric–Gauss–Siedel Method for the Euler and Navier–Stokes Equations. *AIAA Journal*, 26:1025–1026, 1988.

[78] Subhash Saini, Johnny Chang, Robert Hood, and Haoqiang Jin. A Scalability Study of Columbia using the NAS Parallel Benchmarks. Technical Report TP–NAS–06-011, NASA Advanced Supercomputing Division, August 2006.

[79] John V. Becker. The x-15 program in retrospect. `http://www.hq.nasa.gov/office/pao/History/x15lect/cover.html`, December 1968. Presented at the 1st Annual Meeting, Deutsche Gesellschaft f ur Luft- und Raumfahrt, Bonn, Germany.

[80] B. Edney. Anomalous heat transfer and pressure distributions on blunt bodies at hypersonic speeds in the presence of an impinging shock. Technical report, Flygtekniska Försöksanstalten (the Aeronautical Research Institute of Sweden), Stockholm, 1968.

[81] T. Pot, B. Chanetz, M. Lefebvre, and P. Bouchardy. Fundamental study of shock/shock interference in low density flow. 21st International Symposium on Rarefied Gas Dynamics, 1998.

[82] J. N. Moss, T. Pot, B. Chanetz, and M. Lefebvre. DSMC simulation of shock/shock interactions: emphasis on type IV interactions. 22nd International Symposium on Shock Waves, July 1999.

[83] Stephen J. Alter, James J. Reuther, and Ryan D. McDaniel. Development of a Flexible Framework for Hypersonic Navier-Stokes Shuttle Orbiter Meshes. AIAA Paper 2004-2635, June 2004.

[84] National Aeronautics and Space Administration. NASA Dryden Flight Research Center Photo Collection. `http://www.dfrc.nasa.gov/gallery/photo/index.html`.

[85] In-Flight Anomaly Database for STS-1 Through STS-107. `http://www.jsc.nasa.gov/news/columbia/anomaly`, July 2003. Kim Dismukes, Curator.

Vita

Benjamin Shelton Kirk was born in Durham, North Carolina on January 26, 1978. During his career as a graduate student, Ben was a Department of Energy Computational Science Graduate Fellow. Research interests include parallel finite element computations, and the development of object-oriented scientific software in C++. He is an active developer for the `libmesh`[1] open source finite element library, and an application code built over this library was used to obtain the numerical results in this dissertation. He also has a long-term interest in a research or teaching position in academia.

Ben joined the National Aeronautics and Space Administration's Lyndon B. Johnson Space Center in the wake of the *Columbia* tragedy and has subsequently been involved in Space Shuttle Return to Flight, Space Shuttle mission support since STS-114, and is currently leading NASA's analysis efforts in developing the aerothermodynamic design environment for the Project Orion Crew Module.

Permanent address: 301 Clear Creek Meadows Drive
League City, TX 77573

This dissertation was typeset with LaTeX[†] by the author.

[1] `http://libmesh.sourceforge.net`

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.